

APPENDIX

DOWNLOAD AND UNZIP DATASET

1. `!rm -rf Dataset`
2. `!rm Dataset.zip`
3. `!gdown`
`https://drive.google.com/u/0/uc?id=11q9KJhcL3DOQ_2QxxEgSQVxy0CC95v6I`
4. `!7z x /content/Dataset.zip`

IMPORT LIBRARIES

5. `import numpy as np`
6. `import pandas as pd`
7. `import matplotlib.pyplot as plt`
8. `import seaborn as sns`
9. `import math`
10. `import tensorflow as tf`
11. `import warnings`
- 12.
13. `from keras.callbacks import EarlyStopping`
14. `from sklearn import metrics`
15. `from sklearn.preprocessing import MinMaxScaler`
16. `from sklearn.metrics import mean_squared_error`
17. `from sklearn.model_selection import train_test_split`
18. `from sklearn.base import BaseEstimator, RegressorMixin`
19. `from itertools import combinations_with_replacement`
20. `from tensorflow.keras.models import Sequential, Model`
21. `from tensorflow.keras.layers import LSTM, Activation, Dense, Dropout, RepeatVector, Flatten`
22. `from keras.layers.convolutional import Conv1D, MaxPooling1D`
- 23.
24. `%matplotlib inline`
25. `warnings.filterwarnings('ignore')`

IMPORT LIBRARIES

26. `bac = pd.read_csv('/content/Dataset/BAC.csv', parse_dates=['Date'], index_col='Date')`
27. `hdb = pd.read_csv('/content/Dataset/HDB.csv', parse_dates=['Date'], index_col='Date')`
28. `ry = pd.read_csv('/content/Dataset/Ry.csv', parse_dates=['Date'], index_col='Date')`

FUNCTION HEATMAP

29. `def show_heatmap(data):`
30. `fig, ax = plt.subplots(figsize=(15, 15))`
31. `sns.heatmap(data.corr(), linewidth=0.3, cbar_kws={"shrink": .8}, annot=True)`
32. `plt.show()`

VISUALIZE DATA BAC

33. `bac.head(10)`
34. `bac = bac.astype('float64')`

```

35. bac['Volume'] = bac['Volume'].astype('int64')

36. bac.info()

37. bac.describe()

38. bac.replace(np.nan, '0', inplace = True)
39. print("Jumlah data yang duplicate: ", bac.duplicated().sum())
40. print("Jumlah data yang null:")
41. bac.isnull().sum()

42. bac[['Open', 'High', 'Low', 'Close', 'Adj Close']].plot(kind='box')

43. cols = ['Open', 'High', 'Low', 'Close', 'Volume']
44. plt.figure(figsize=(16, 8))
45. axes = bac[cols].plot(figsize=(11, 9), subplots = True)
46. plt.show()

47. show_heatmap(bac)

48. bac['HL_PCT'] = (bac['High'] - bac['Low']) / bac['Low'] * 100.0
49. bac['PCT_change'] = (bac['Close'] - bac['Open']) / bac['Open'] * 100.0

50. bac['Open:30 days rolling']=bac['Open'].rolling(30).mean()
51. bac['High:30 days rolling']=bac['High'].rolling(30).mean()
52. bac['Low:30 days rolling']=bac['Low'].rolling(30).mean()
53. bac['Close:30 days rolling']=bac['Close'].rolling(30).mean()
54. bac['Adj Close:30 days rolling']=bac['Adj Close'].rolling(30).mean()
55. bac['Volume:30 days rolling']=bac['Volume'].rolling(30).mean()
56.
57. bac[['Open', 'Open:30 days rolling']].plot(figsize=(12,5))
58. bac[['High', 'High:30 days rolling']].plot(figsize=(12,5))
59. bac[['Low', 'Low:30 days rolling']].plot(figsize=(12,5))
60. bac[['Close', 'Close:30 days rolling']].plot(figsize=(12,5))
61. bac[['Adj Close', 'Adj Close:30 days rolling']].plot(figsize=(12,5))
62. bac[['Volume', 'Volume:30 days rolling']].plot(figsize=(12,5))

63. bac['Open'].resample(rule='M').max().plot(kind='bar', figsize=(15,6))

64. bac['delta_open_close_day_before %']=(bac['Open']-
    bac['Close'].shift(1))/bac['Close'].shift(1)*100
65. order = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday"]
66. box_plot = sns.boxplot(x=pd.to_datetime(bac.index).day_name(),
    y=bac['delta_open_close_day_before %'], order=order)
67.
68. ax = box_plot.axes
69. lines = ax.get_lines()
70. categories = ax.get_xticks()
71.
72. for cat in categories:
73.     y = round(lines[4+cat*6].get_ydata()[0],1)
74.
75.     ax.text(
76.         cat,
77.         y,
78.         f'{y}',
79.         ha='center',
80.         va='center',
81.         fontweight='bold',

```

```

82.         size=10,
83.         color='white',
84.         bbox=dict(facecolor='#445A64'))
85.
86. box_plot.figure.tight_layout()

```

VISUALIZE DATA HDB

```

87. hdb.head(10)

88. hdb = hdb.astype('float64')

89. hdb['Volume'] = hdb['Volume'].astype('int64')

90. hdb.info()

91. hdb.describe()

92. hdb.replace(np.nan, '0', inplace = True)
93. print("Jumlah data yang duplicate: ", hdb.duplicated().sum())
94. print("Jumlah data yang null:")
95. hdb.isnull().sum()

96. hdb[['Open', 'High', 'Low', 'Close', 'Adj Close']].plot(kind='box')

97. cols = ['Open', 'High', 'Low', 'Close', 'Volume']
98. plt.figure(figsize=(16, 8))
99. axes = hdb[cols].plot(figsize=(11, 9), subplots = True)
100. plt.show()

101. show_heatmap(hdb)

102. hdb['HL_PCT'] = (hdb['High'] - hdb['Low']) / hdb['Low'] * 100.0
103. hdb['PCT_change'] = (hdb['Close'] - hdb['Open']) / hdb['Open'] * 100.0

104. hdb['Open:30 days rolling']=hdb['Open'].rolling(30).mean()
105. hdb['High:30 days rolling']=hdb['High'].rolling(30).mean()
106. hdb['Low:30 days rolling']=hdb['Low'].rolling(30).mean()
107. hdb['Close:30 days rolling']=hdb['Close'].rolling(30).mean()
108. hdb['Adj Close:30 days rolling']=hdb['Adj Close'].rolling(30).mean()
109. hdb['Volume:30 days rolling']=hdb['Volume'].rolling(30).mean()
110.
111. hdb[['Open', 'Open:30 days rolling']].plot(figsize=(12,5))
112. hdb[['High', 'High:30 days rolling']].plot(figsize=(12,5))
113. hdb[['Low', 'Low:30 days rolling']].plot(figsize=(12,5))
114. hdb[['Close', 'Close:30 days rolling']].plot(figsize=(12,5))
115. hdb[['Adj Close', 'Adj Close:30 days rolling']].plot(figsize=(12,5))
116. hdb[['Volume', 'Volume:30 days rolling']].plot(figsize=(12,5))

117. hdb['Open'].resample(rule='M').max().plot(kind='bar', figsize=(15,6))

118. hdb['delta_open_close_day_before_%']=(hdb['Open']-
      hdb['Close'].shift(1))/hdb['Close'].shift(1)*100
119. order = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday"]
120. box_plot = sns.boxplot(x=pd.to_datetime(hdb.index).day_name(),
      y=hdb['delta_open_close_day_before_%'], order=order)
121.
122. ax = box_plot.axes

```

```

123.lines = ax.get_lines()
124.categories = ax.get_xticks()
125.
126.for cat in categories:
127.    y = round(lines[4+cat*6].get_ydata()[0],1)
128.
129.    ax.text(
130.        cat,
131.        y,
132.        f'{y}',
133.        ha='center',
134.        va='center',
135.        fontweight='bold',
136.        size=10,
137.        color='white',
138.        bbox=dict(facecolor='#445A64'))
139.
140.box_plot.figure.tight_layout()

```

VISUALIZE DATA RY

```

141.ry.head(10)

142.ry = ry.astype('float64')

143.ry['Volume'] = ry['Volume'].astype('int64')

144.ry.info()

145.ry.describe()

146.ry.replace(np.nan,'0',inplace = True)
147.print("Jumlah data yang duplicate: ", ry.duplicated().sum())
148.print("Jumlah data yang null:")
149.ry.isnull().sum()

150.ry[['Open', 'High', 'Low', 'Close', 'Adj Close']].plot(kind='box')

151.cols = ['Open', 'High', 'Low', 'Close', 'Volume']
152.plt.figure(figsize=(16, 8))
153.axes = ry[cols].plot(figsize=(11, 9), subplots = True)
154.plt.show()

155.show_heatmap(ry)

156.ry['HL_PCT'] = (ry['High'] - ry['Low']) / ry['Low'] * 100.0
157.ry['PCT_change'] = (ry['Close'] - ry['Open']) / ry['Open'] * 100.0

158.ry['Open:30 days rolling']=ry['Open'].rolling(30).mean()
159.ry['High:30 days rolling']=ry['High'].rolling(30).mean()
160.ry['Low:30 days rolling']=ry['Low'].rolling(30).mean()
161.ry['Close:30 days rolling']=ry['Close'].rolling(30).mean()
162.ry['Adj Close:30 days rolling']=ry['Adj Close'].rolling(30).mean()
163.ry['Volume:30 days rolling']=ry['Volume'].rolling(30).mean()
164.
165.ry[['Open','Open:30 days rolling']].plot(figsize=(12,5))
166.ry[['High','High:30 days rolling']].plot(figsize=(12,5))
167.ry[['Low','Low:30 days rolling']].plot(figsize=(12,5))

```

```

168.ry[['Close','Close:30 days rolling']].plot(figsize=(12,5))
169.ry[['Adj Close','Adj Close:30 days rolling']].plot(figsize=(12,5))
170.ry[['Volume','Volume:30 days rolling']].plot(figsize=(12,5))

171.ry['Open'].resample(rule='M').max().plot(kind='bar',figsize=(15,6))

172.ry['delta_open_close_day_before_%']=(ry['Open']-
ry['Close'].shift(1))/ry['Close'].shift(1)*100
173.order = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday"]
174.box_plot = sns.boxplot(x=pd.to_datetime(ry.index).day_name(),
y=ry['delta_open_close_day_before_%'], order=order)
175.
176.ax = box_plot.axes
177.lines = ax.get_lines()
178.categories = ax.get_xticks()
179.
180.for cat in categories:
181.    y = round(lines[4+cat*6].get_ydata()[0],1)
182.
183.    ax.text(
184.        cat,
185.        y,
186.        f'{y}',
187.        ha='center',
188.        va='center',
189.        fontweight='bold',
190.        size=10,
191.        color='white',
192.        bbox=dict(facecolor='#445A64'))
193.
194.box_plot.figure.tight_layout()

```

FUNCTION EVALUATION

```

195.def rmse(y_test, y_pred):
196.    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
197.    return rmse

198.def mae(y_test, y_pred):
199.    mae = metrics.mean_absolute_error(y_test, y_pred)
200.    return mae

201.def mape(y_test, y_pred):
202.    mape = np.mean(np.abs((y_test - y_pred)/y_test))*100
203.    return mape

```

FEATURE SCALING

```

204.bac = bac[['HL_PCT', 'PCT_change', 'Adj Close','Volume']]
205.bac_forecast_out = int(math.ceil(0.1 * len(bac))) # forecasting out
10% of the entire dataset
206.bac['label'] = bac['Adj Close'].shift(-bac_forecast_out)

207.hdb = hdb[['HL_PCT', 'PCT_change', 'Adj Close','Volume']]
208.hdb_forecast_out = int(math.ceil(0.1 * len(hdb))) # forecasting out
10% of the entire dataset
209.hdb['label'] = hdb['Adj Close'].shift(-hdb_forecast_out)

210.ry = ry[['HL_PCT', 'PCT_change', 'Adj Close','Volume']]

```

```

211.ry_forecast_out = int(math.ceil(0.1 * len(ry))) # forecasting out 10%
    of the entire dataset
212.ry['label'] = ry['Adj Close'].shift(-ry_forecast_out)

213.sc= MinMaxScaler(feature_range=(0,1))

214.bac_X = np.array(bac.drop(['label'], 1))
215.sc.fit(bac_X)
216.bac_X = sc.transform(bac_X)

217.hdb_X = np.array(hdb.drop(['label'], 1))
218.sc.fit(hdb_X)
219.hdb_X = sc.transform(hdb_X)

220.ry_X = np.array(ry.drop(['label'], 1))
221.sc.fit(ry_X)
222.ry_X = sc.transform(ry_X)

```

PREPARING DATA

```

223.X_bac_Predictions = bac_X[-bac_forecast_out:]
224.X_bac = bac_X[:-bac_forecast_out]
225.bac.dropna(inplace=True)
226.y_bac = np.array(bac['label'])

227.X_hdb_Predictions = hdb_X[-hdb_forecast_out:]
228.X_hdb = hdb_X[:-hdb_forecast_out]
229.hdb.dropna(inplace=True)
230.y_hdb = np.array(hdb['label'])

231.# Preparing Data ry For Training
232.X_ry_Predictions = ry_X[-hdb_forecast_out:]
233.X_ry = ry_X[:-ry_forecast_out]
234.ry.dropna(inplace=True)
235.y_ry = np.array(ry['label'])

```

SPLIT TRAIN DATA AND TEST DATA

```

236.bac20_X_train, bac20_X_test, bac20_y_train, bac20_y_test =
    train_test_split(X_bac, y_bac, test_size=0.2, random_state=42)
237.hdb20_X_train, hdb20_X_test, hdb20_y_train, hdb20_y_test =
    train_test_split(X_hdb, y_hdb, test_size=0.2, random_state=42)
238.ry20_X_train, ry20_X_test, ry20_y_train, ry20_y_test =
    train_test_split(X_ry, y_ry, test_size=0.2, random_state=42)

239.bac40_X_train, bac40_X_test, bac40_y_train, bac40_y_test =
    train_test_split(X_bac, y_bac, test_size=0.4, random_state=42)
240.hdb40_X_train, hdb40_X_test, hdb40_y_train, hdb40_y_test =
    train_test_split(X_hdb, y_hdb, test_size=0.4, random_state=42)
241.ry40_X_train, ry40_X_test, ry40_y_train, ry40_y_test =
    train_test_split(X_ry, y_ry, test_size=0.4, random_state=42)

242.bac50_X_train, bac50_X_test, bac50_y_train, bac50_y_test =
    train_test_split(X_bac, y_bac, test_size=0.5, random_state=42)
243.hdb50_X_train, hdb50_X_test, hdb50_y_train, hdb50_y_test =
    train_test_split(X_hdb, y_hdb, test_size=0.5, random_state=42)
244.ry50_X_train, ry50_X_test, ry50_y_train, ry50_y_test =
    train_test_split(X_ry, y_ry, test_size=0.5, random_state=42)

245.bac60_X_train, bac60_X_test, bac60_y_train, bac60_y_test =
    train_test_split(X_bac, y_bac, test_size=0.6, random_state=42)

```

```

246.hdb60_X_train, hdb60_X_test, hdb60_y_train, hdb60_y_test =
    train_test_split(X_hdb, y_hdb, test_size=0.6, random_state=42)
247.ry60_X_train, ry60_X_test, ry60_y_train, ry60_y_test =
    train_test_split(X_ry, y_ry, test_size=0.6, random_state=42)

248.bac80_X_train, bac80_X_test, bac80_y_train, bac80_y_test =
    train_test_split(X_bac, y_bac, test_size=0.8, random_state=42)
249.hdb80_X_train, hdb80_X_test, hdb80_y_train, hdb80_y_test =
    train_test_split(X_hdb, y_hdb, test_size=0.8, random_state=42)
250.ry80_X_train, ry80_X_test, ry80_y_train, ry80_y_test =
    train_test_split(X_ry, y_ry, test_size=0.8, random_state=42)

```

CLASS REGRESSION

```

251.class Regression(object):
252.     def __init__(self, n_iterations, learning_rate):
253.         self.m = None
254.         self.n = None
255.         self.w = None
256.         self.b = None
257.         self.n_iterations = n_iterations
258.         self.learning_rate = learning_rate
259.
260.     def cost_function(self, y, y_pred):
261.         return (1 / (2*self.m)) * np.sum(np.square(y_pred - y)) +
            self.regularization(self.w)
262.
263.     def hypothesis(self, weights, X):
264.         return np.dot(X, weights)
265.
266.     def initialize_weights(self, n_features):
267.         limit = 1 / math.sqrt(n_features)
268.         self.w = np.random.uniform(-limit, limit, (n_features, ))
269.
270.     def fit(self, X, y):
271.         X = np.insert(X, 0, 1, axis=1)
272.         self.training_errors = []
273.         self.initialize_weights(n_features=X.shape[1])
274.
275.         self.m = X.shape[0]
276.         self.n = X.shape[1]
277.
278.         self.b = 0
279.
280.         for i in range(self.n_iterations):
281.             y_pred = self.hypothesis(self.w, X)
282.             cost = self.cost_function(y, y_pred)
283.             self.training_errors.append(cost)
284.             grad_w = -(y - y_pred).dot(X) +
                self.regularization.grad(self.w)
285.             self.w = self.w - self.learning_rate * grad_w
286.
287.     def predict(self, X):
288.         X = np.insert(X, 0, 1, axis=1)
289.         y_pred = self.hypothesis(self.w, X)
290.         return y_pred

```

BUILD LINEAR REGRESSION

```

291.class LinearRegression(Regression):

```

```

292.     def __init__(self, n_iterations=100, learning_rate=0.001,
        gradient_descent=True):
293.         self.gradient_descent = gradient_descent
294.         self.regularization = lambda x: 0
295.         self.regularization.grad = lambda x: 0
296.         super(LinearRegression,
        self).__init__(n_iterations=n_iterations,
297. learning_rate=learning_rate)
298.     def fit(self, X, y): w
299.         if not self.gradient_descent:
300.             X = np.insert(X, 0, 1, axis=1)
301.             U, S, V = np.linalg.svd(X.T.dot(X))
302.             S = np.diag(S)
303.             X_sq_reg_inv = V.dot(np.linalg.pinv(S)).dot(U.T)
304.             self.w = X_sq_reg_inv.dot(X.T).dot(y)
305.         else:
306.             super(LinearRegression, self).fit(X, y)

```

BUILD LASSO REGRESSION

```

307. def normalize(X, axis=-1, order=2):
308.     l2 = np.atleast_1d(np.linalg.norm(X, order, axis))
309.     l2[l2 == 0] = 1
310.     return X / np.expand_dims(l2, axis)

311. def polynomial_features(X, lamda):
312.     n_samples, n_features = np.shape(X)
313.     def index_combinations():
314.         combs = [combinations_with_replacement(range(n_features), i)
        for i in range(0, lamda + 1)]
315.         flat_combs = [item for sublist in combs for item in sublist]
316.         return flat_combs
317.
318.     combinations = index_combinations()
319.     n_output_features = len(combinations)
320.     X_new = np.empty((n_samples, n_output_features))
321.
322.     for i, index_combs in enumerate(combinations):
323.         X_new[:, i] = np.prod(X[:, index_combs], axis=1)
324.
325.     return X_new

326. class l1_regularization():
327.     def __init__(self, alpha):
328.         self.alpha = alpha
329.
330.     def __call__(self, w):
331.         return self.alpha * np.linalg.norm(w)
332.
333.     def grad(self, w):
334.         return self.alpha * np.sign(w)

335. class LassoRegression(Regression):
336.     def __init__(self, lamda, n_iterations=3000, learning_rate=0.01):
337.         self.lamda = lamda
338.         self.regularization = l1_regularization(alpha=0.05)
339.         super(LassoRegression, self).__init__(n_iterations,
        learning_rate)
340.
341.     def fit(self, X, y):

```



```

342.         X = normalize(X)
343.         super(LassoRegression, self).fit(X, y)
344.
345.     def predict(self, X):
346.         X = normalize(X)
347.         return super(LassoRegression, self).predict(X)

```

BUILD LSSVM

```

348. class LSSVMRegression(BaseEstimator, RegressorMixin):
349.     def __init__(self, gamma: float = 1.0, kernel: str = None, c: float
      = 1.0, d: float = 2, sigma: float = 1.0):
350.         self.gamma = gamma
351.         self.c = c
352.         self.d = d
353.         self.sigma = sigma
354.         self.kernel = 'rbf'
355.
356.         params = dict()
357.         params['sigma'] = sigma
358.
359.         self.kernel_ = LSSVMRegression.__set_kernel(self.kernel,
      **params)
360.         self.x = None
361.         self.y = None
362.         self.coef_ = None
363.         self.intercept_ = None
364.
365.     def get_params(self, deep=True):
366.         return {"c": self.c, "d": self.d, "gamma": self.gamma, "kernel":
      self.kernel, "sigma": self.sigma}
367.
368.     def set_params(self, **parameters):
369.         for parameter, value in parameters.items():
370.             setattr(self, parameter, value)
371.         params = dict()
372.         params['sigma'] = self.sigma
373.         self.kernel_ = LSSVMRegression.rbf(self.x, self.y, **params)
374.         return self
375.
376.     def set_attributes(self, **parameters):
377.         for param, value in parameters.items():
378.             if param == 'intercept_':
379.                 self.intercept_ = value
380.             elif param == 'coef_':
381.                 self.coef_ = value
382.             elif param == 'support_':
383.                 self.x = value
384.
385.     def __set_kernel(name: str, **params):
386.         def rbf(xi, xj, sigma=params.get('sigma', 0.5)):
387.             from scipy.spatial.distance import cdist
388.             if (xi.ndim == 2 and xi.ndim == xj.ndim):
389.                 return np.exp(-(cdist(xi, xj,
      metric='sqeuclidean'))/(2*(sigma**2)))
390.             else:
391.                 message = "The rbf kernel is not suited for arrays with rank
      >2"
392.                 raise Exception(message)
393.

```

```

394.     kernels = {'rbf': rbf}
395.     return kernels[name]
396.
397. def __OptimizeParams(self):
398.     Omega = self.kernel_(self.x, self.x)
399.     Ones = np.array([[1]]*len(self.y))
400.
401.     A_dag = np.linalg.pinv(np.block([
402.         [0,          Ones.T          ],
403.         [Ones,  Omega + self.gamma**-1 * np.identity(len(self.y))]
404.     ]))
405.     B = np.concatenate((np.array([0]), self.y), axis=None)
406.
407.     solution = np.dot(A_dag, B)
408.     self.intercept_ = solution[0]
409.     self.coef_      = solution[1:]
410.
411. def fit(self, X: np.ndarray, y: np.ndarray):
412.     if isinstance(X, (pd.DataFrame, pd.Series)):
413.         Xloc = X.to_numpy()
414.     else:
415.         Xloc = X
416.
417.     if isinstance(y, (pd.DataFrame, pd.Series)):
418.         yloc = y.to_numpy()
419.     else:
420.         yloc = y
421.
422.     if (Xloc.ndim == 2) and (yloc.ndim == 1):
423.         self.x = Xloc
424.         self.y = yloc
425.         self.__OptimizeParams()
426.     else:
427.         message = "The fit procedure requires a 2D numpy array of
428.         features "\
429.         "and 1D array of targets"
430.         raise Exception(message)
431.
432. def predict(self, X: np.ndarray)->np.ndarray:
433.     Ker = self.kernel_(X, self.x)
434.     Y = np.dot(self.coef_, Ker.T) + self.intercept_
435.     return Y

```

BUILD LSTM FOR DATASET BAC

```

435.model_bac_20= Sequential()
436.model_bac_20.add(LSTM(units=100,return_sequences=True,
437.     activation='relu', input_shape=(bac20_X_train.shape[1], 1)))
438.model_bac_20.add(Dropout(rate=0.2))
439.model_bac_20.add(LSTM(units=100,return_sequences=True,
440.     activation='relu'))
441.model_bac_20.add(Dropout(rate=0.01))
442.model_bac_20.add(LSTM(units=100,return_sequences=True,
443.     activation='relu'))
444.model_bac_20.add(Dropout(rate=0.002))
445.model_bac_20.add(LSTM(units=100, return_sequences=True,
446.     activation='relu'))
447.model_bac_20.add(Dropout(rate=0.02))
448.model_bac_20.add(LSTM(units=100, activation='relu'))
449.model_bac_20.add(Dropout(rate=0.02))

```

```

446.model_bac_20.add(Dense(units=100))
447.model_bac_20.compile(loss='mse', optimizer='adam', metrics=['mse',
'mae', 'mape'])

448.model_bac_20.summary()

449.dot_img_file = '/tmp/model_lstm.png'
450.tf.keras.utils.plot_model(model_bac_20, to_file=dot_img_file,
show_shapes=True)

451.model_bac_40= Sequential()
452.model_bac_40.add(LSTM(units=100,return_sequences=True,
activation='relu', input_shape=(bac40_X_train.shape[1], 1)))
453.model_bac_40.add(Dropout(rate=0.2))
454.model_bac_40.add(LSTM(units=100,return_sequences=True,
activation='relu'))
455.model_bac_40.add(Dropout(rate=0.01))
456.model_bac_40.add(LSTM(units=100,return_sequences=True,
activation='relu'))
457.model_bac_40.add(Dropout(rate=0.002))
458.model_bac_40.add(LSTM(units=100,return_sequences=True,
activation='relu'))
459.model_bac_40.add(Dropout(rate=0.02))
460.model_bac_40.add(LSTM(units=100, activation='relu'))
461.model_bac_40.add(Dropout(rate=0.02))
462.model_bac_40.add(Dense(units=100))
463.model_bac_40.compile(loss='mse', optimizer='adam', metrics=['mse',
'mae', 'mape'])

464.model_bac_50= Sequential()
465.model_bac_50.add(LSTM(units=100,return_sequences=True,
activation='relu', input_shape=(bac50_X_train.shape[1], 1)))
466.model_bac_50.add(Dropout(rate=0.2))
467.model_bac_50.add(LSTM(units=100,return_sequences=True,
activation='relu'))
468.model_bac_50.add(Dropout(rate=0.01))
469.model_bac_50.add(LSTM(units=100,return_sequences=True,
activation='relu'))
470.model_bac_50.add(Dropout(rate=0.002))
471.model_bac_50.add(LSTM(units=100,return_sequences=True,
activation='relu'))
472.model_bac_50.add(Dropout(rate=0.02))
473.model_bac_50.add(LSTM(units=100, activation='relu'))
474.model_bac_50.add(Dropout(rate=0.02))
475.model_bac_50.add(Dense(units=100))
476.model_bac_50.compile(loss='mse', optimizer='adam', metrics=['mse',
'mae', 'mape'])

477.model_bac_60= Sequential()
478.model_bac_60.add(LSTM(units=100,return_sequences=True,
activation='relu', input_shape=(bac60_X_train.shape[1], 1)))
479.model_bac_60.add(Dropout(rate=0.2))
480.model_bac_60.add(LSTM(units=100,return_sequences=True,
activation='relu'))
481.model_bac_60.add(Dropout(rate=0.01))
482.model_bac_60.add(LSTM(units=100,return_sequences=True,
activation='relu'))
483.model_bac_60.add(Dropout(rate=0.002))

```

```

484.model_bac_60.add(LSTM(units=100,return_sequences=True,
    activation='relu'))
485.model_bac_60.add(Dropout(rate=0.02))
486.model_bac_60.add(LSTM(units=100, activation='relu'))
487.model_bac_60.add(Dropout(rate=0.02))
488.model_bac_60.add(Dense(units=100))
489.model_bac_60.compile(loss='mse', optimizer='adam', metrics=['mse',
    'mae', 'mape'])

490.model_bac_80= Sequential()
491.model_bac_80.add(LSTM(units=100,return_sequences=True,
    activation='relu', input_shape=(bac80_X_train.shape[1], 1)))
492.model_bac_80.add(Dropout(rate=0.2))
493.model_bac_80.add(LSTM(units=100,return_sequences=True,
    activation='relu'))
494.model_bac_80.add(Dropout(rate=0.01))
495.model_bac_80.add(LSTM(units=100,return_sequences=True,
    activation='relu'))
496.model_bac_80.add(Dropout(rate=0.002))
497.model_bac_80.add(LSTM(units=100,return_sequences=True,
    activation='relu'))
498.model_bac_80.add(Dropout(rate=0.02))
499.model_bac_80.add(LSTM(units=100, activation='relu'))
500.model_bac_80.add(Dropout(rate=0.02))
501.model_bac_80.add(Dense(units=100))
502.model_bac_80.compile(loss='mse', optimizer='adam', metrics=['mse',
    'mae', 'mape'])

```

BUILD LSTM FOR DATASET HDB

```

503.model_hdb_20= Sequential()
504.model_hdb_20.add(LSTM(units=100,return_sequences=True,
    activation='relu', input_shape=(hdb20_X_train.shape[1], 1)))
505.model_hdb_20.add(Dropout(rate=0.2))
506.model_hdb_20.add(LSTM(units=100,return_sequences=True,
    activation='relu'))
507.model_hdb_20.add(Dropout(rate=0.01))
508.model_hdb_20.add(LSTM(units=100,return_sequences=True,
    activation='relu'))
509.model_hdb_20.add(Dropout(rate=0.002))
510.model_hdb_20.add(LSTM(units=100,return_sequences=True,
    activation='relu'))
511.model_hdb_20.add(Dropout(rate=0.02))
512.model_hdb_20.add(LSTM(units=100, activation='relu'))
513.model_hdb_20.add(Dropout(rate=0.02))
514.model_hdb_20.add(Dense(units=100))
515.model_hdb_20.compile(loss='mse', optimizer='adam', metrics=['mse',
    'mae', 'mape'])

516.model_hdb_40= Sequential()
517.model_hdb_40.add(LSTM(units=100,return_sequences=True,
    activation='relu', input_shape=(hdb40_X_train.shape[1], 1)))
518.model_hdb_40.add(Dropout(rate=0.2))
519.model_hdb_40.add(LSTM(units=100,return_sequences=True,
    activation='relu'))
520.model_hdb_40.add(Dropout(rate=0.01))
521.model_hdb_40.add(LSTM(units=100,return_sequences=True,
    activation='relu'))
522.model_hdb_40.add(Dropout(rate=0.002))

```

```

523.model_hdb_40.add(LSTM(units=100,return_sequences=True,
    activation='relu'))
524.model_hdb_40.add(Dropout(rate=0.02))
525.model_hdb_40.add(LSTM(units=100, activation='relu'))
526.model_hdb_40.add(Dropout(rate=0.02))
527.model_hdb_40.add(Dense(units=100))
528.model_hdb_40.compile(loss='mse', optimizer='adam', metrics=['mse',
    'mae', 'mape'])

529.model_hdb_50= Sequential()
530.model_hdb_50.add(LSTM(units=100,return_sequences=True,
    activation='relu', input_shape=(hdb50_X_train.shape[1], 1)))
531.model_hdb_50.add(Dropout(rate=0.2))
532.model_hdb_50.add(LSTM(units=100,return_sequences=True,
    activation='relu'))
533.model_hdb_50.add(Dropout(rate=0.01))
534.model_hdb_50.add(LSTM(units=100,return_sequences=True,
    activation='relu'))
535.model_hdb_50.add(Dropout(rate=0.002))
536.model_hdb_50.add(LSTM(units=100,return_sequences=True,
    activation='relu'))
537.model_hdb_50.add(Dropout(rate=0.02))
538.model_hdb_50.add(LSTM(units=100, activation='relu'))
539.model_hdb_50.add(Dropout(rate=0.02))
540.model_hdb_50.add(Dense(units=100))
541.model_hdb_50.compile(loss='mse', optimizer='adam', metrics=['mse',
    'mae', 'mape'])

542.model_hdb_60= Sequential()
543.model_hdb_60.add(LSTM(units=100,return_sequences=True,
    activation='relu', input_shape=(hdb60_X_train.shape[1], 1)))
544.model_hdb_60.add(Dropout(rate=0.2))
545.model_hdb_60.add(LSTM(units=100,return_sequences=True,
    activation='relu'))
546.model_hdb_60.add(Dropout(rate=0.01))
547.model_hdb_60.add(LSTM(units=100,return_sequences=True,
    activation='relu'))
548.model_hdb_60.add(Dropout(rate=0.002))
549.model_hdb_60.add(LSTM(units=100,return_sequences=True,
    activation='relu'))
550.model_hdb_60.add(Dropout(rate=0.02))
551.model_hdb_60.add(LSTM(units=100, activation='relu'))
552.model_hdb_60.add(Dropout(rate=0.02))
553.model_hdb_60.add(Dense(units=100))
554.model_hdb_60.compile(loss='mse', optimizer='adam', metrics=['mse',
    'mae', 'mape'])

555.model_hdb_80= Sequential()
556.model_hdb_80.add(LSTM(units=100,return_sequences=True,
    activation='relu', input_shape=(hdb80_X_train.shape[1], 1)))
557.model_hdb_80.add(Dropout(rate=0.2))
558.model_hdb_80.add(LSTM(units=100,return_sequences=True,
    activation='relu'))
559.model_hdb_80.add(Dropout(rate=0.01))
560.model_hdb_80.add(LSTM(units=100,return_sequences=True,
    activation='relu'))
561.model_hdb_80.add(Dropout(rate=0.002))
562.model_hdb_80.add(LSTM(units=100,return_sequences=True,
    activation='relu'))

```

```

563.model_hdb_80.add(Dropout(rate=0.02))
564.model_hdb_80.add(LSTM(units=100, activation='relu'))
565.model_hdb_80.add(Dropout(rate=0.02))
566.model_hdb_80.add(Dense(units=100))
567.model_hdb_80.compile(loss='mse', optimizer='adam', metrics=['mse',
'mae', 'mape'])

```

BUILD LSTM FOR DATASET RY

```

568.model_ry_20= Sequential()
569.model_ry_20.add(LSTM(units=100,return_sequences=True,
activation='relu', input_shape=(ry20_X_train.shape[1], 1)))
570.model_ry_20.add(Dropout(rate=0.2))
571.model_ry_20.add(LSTM(units=100,return_sequences=True,
activation='relu'))
572.model_ry_20.add(Dropout(rate=0.01))
573.model_ry_20.add(LSTM(units=100,return_sequences=True,
activation='relu'))
574.model_ry_20.add(Dropout(rate=0.002))
575.model_ry_20.add(LSTM(units=100,return_sequences=True,
activation='relu'))
576.model_ry_20.add(Dropout(rate=0.02))
577.model_ry_20.add(LSTM(units=100, activation='relu'))
578.model_ry_20.add(Dropout(rate=0.02))
579.model_ry_20.add(Dense(units=100))
580.model_ry_20.compile(loss='mse', optimizer='adam', metrics=['mse',
'mae', 'mape'])

581.model_ry_40= Sequential()
582.model_ry_40.add(LSTM(units=100,return_sequences=True,
activation='relu', input_shape=(ry40_X_train.shape[1], 1)))
583.model_ry_40.add(Dropout(rate=0.2))
584.model_ry_40.add(LSTM(units=100,return_sequences=True,
activation='relu'))
585.model_ry_40.add(Dropout(rate=0.01))
586.model_ry_40.add(LSTM(units=100,return_sequences=True,
activation='relu'))
587.model_ry_40.add(Dropout(rate=0.002))
588.model_ry_40.add(LSTM(units=100,return_sequences=True,
activation='relu'))
589.model_ry_40.add(Dropout(rate=0.02))
590.model_ry_40.add(LSTM(units=100, activation='relu'))
591.model_ry_40.add(Dropout(rate=0.02))
592.model_ry_40.add(Dense(units=100))
593.model_ry_40.compile(loss='mse', optimizer='adam', metrics=['mse',
'mae', 'mape'])

594.model_ry_50= Sequential()
595.model_ry_50.add(LSTM(units=100,return_sequences=True,
activation='relu', input_shape=(ry50_X_train.shape[1], 1)))
596.model_ry_50.add(Dropout(rate=0.2))
597.model_ry_50.add(LSTM(units=100,return_sequences=True,
activation='relu'))
598.model_ry_50.add(Dropout(rate=0.01))
599.model_ry_50.add(LSTM(units=100,return_sequences=True,
activation='relu'))
600.model_ry_50.add(Dropout(rate=0.002))
601.model_ry_50.add(LSTM(units=100,return_sequences=True,
activation='relu'))
602.model_ry_50.add(Dropout(rate=0.02))

```

```

603.model_ry_50.add(LSTM(units=100, activation='relu'))
604.model_ry_50.add(Dropout(rate=0.02))
605.model_ry_50.add(Dense(units=100))
606.model_ry_50.compile(loss='mse', optimizer='adam', metrics=['mse',
'mae', 'mape'])

607.model_ry_60= Sequential()
608.model_ry_60.add(LSTM(units=100,return_sequences=True,
activation='relu', input_shape=(ry60_X_train.shape[1], 1)))
609.model_ry_60.add(Dropout(rate=0.2))
610.model_ry_60.add(LSTM(units=100,return_sequences=True,
activation='relu'))
611.model_ry_60.add(Dropout(rate=0.01))
612.model_ry_60.add(LSTM(units=100,return_sequences=True,
activation='relu'))
613.model_ry_60.add(Dropout(rate=0.002))
614.model_ry_60.add(LSTM(units=100,return_sequences=True,
activation='relu'))
615.model_ry_60.add(Dropout(rate=0.02))
616.model_ry_60.add(LSTM(units=100, activation='relu'))
617.model_ry_60.add(Dropout(rate=0.02))
618.model_ry_60.add(Dense(units=100))
619.model_ry_60.compile(loss='mse', optimizer='adam', metrics=['mse',
'mae', 'mape'])

620.model_ry_80= Sequential()
621.model_ry_80.add(LSTM(units=100,return_sequences=True,
activation='relu', input_shape=(ry80_X_train.shape[1], 1)))
622.model_ry_80.add(Dropout(rate=0.2))
623.model_ry_80.add(LSTM(units=100,return_sequences=True,
activation='relu'))
624.model_ry_80.add(Dropout(rate=0.01))
625.model_ry_80.add(LSTM(units=100,return_sequences=True,
activation='relu'))
626.model_ry_80.add(Dropout(rate=0.002))
627.model_ry_80.add(LSTM(units=100,return_sequences=True,
activation='relu'))
628.model_ry_80.add(Dropout(rate=0.02))
629.model_ry_80.add(LSTM(units=100, activation='relu'))
630.model_ry_80.add(Dropout(rate=0.02))
631.model_ry_80.add(Dense(units=100))
632.model_ry_80.compile(loss='mse', optimizer='adam', metrics=['mse',
'mae', 'mape'])

```

BUILD CNN FOR DATASET BAC

```

633.cnn_model_bac_20 = Sequential()
634.cnn_model_bac_20.add(Conv1D(filters=64, kernel_size=2,
activation='relu', input_shape=(bac20_X_train.shape[1], 1)))
635.cnn_model_bac_20.add(MaxPooling1D(pool_size=2))
636.cnn_model_bac_20.add(Conv1D(filters=32, kernel_size=1,
activation='relu'))
637.cnn_model_bac_20.add(MaxPooling1D(pool_size=1))
638.cnn_model_bac_20.add(Conv1D(filters=64, kernel_size=1,
activation='relu'))
639.cnn_model_bac_20.add(MaxPooling1D(pool_size=1))
640.cnn_model_bac_20.add(Conv1D(filters=32, kernel_size=1,
activation='relu'))
641.cnn_model_bac_20.add(MaxPooling1D(pool_size=1))

```

```

642.cnn_model_bac_20.add(Conv1D(filters=64,                kernel_size=1,
    activation='relu'))
643.cnn_model_bac_20.add(MaxPooling1D(pool_size=1))
644.cnn_model_bac_20.add(Flatten())
645.cnn_model_bac_20.add(Dense(50, activation='relu'))
646.cnn_model_bac_20.add(Dense(1))
647.cnn_model_bac_20.compile(loss='mse',                optimizer='adam',
    metrics=['mse', 'mae', 'mape'])

648.cnn_model_bac_20.summary()

649.dot_img_file = '/tmp/model_cnn.png'
650.tf.keras.utils.plot_model(cnn_model_bac_20,        to_file=dot_img_file,
    show_shapes=True)

651.cnn_model_bac_40 = Sequential()
652.cnn_model_bac_40.add(Conv1D(filters=64,                kernel_size=2,
    activation='relu', input_shape=(bac40_X_train.shape[1], 1)))
653.cnn_model_bac_40.add(MaxPooling1D(pool_size=2))
654.cnn_model_bac_40.add(Conv1D(filters=32,                kernel_size=1,
    activation='relu'))
655.cnn_model_bac_40.add(MaxPooling1D(pool_size=1))
656.cnn_model_bac_40.add(Conv1D(filters=64,                kernel_size=1,
    activation='relu'))
657.cnn_model_bac_40.add(MaxPooling1D(pool_size=1))
658.cnn_model_bac_40.add(Conv1D(filters=32,                kernel_size=1,
    activation='relu'))
659.cnn_model_bac_40.add(MaxPooling1D(pool_size=1))
660.cnn_model_bac_40.add(Conv1D(filters=64,                kernel_size=1,
    activation='relu'))
661.cnn_model_bac_40.add(MaxPooling1D(pool_size=1))
662.cnn_model_bac_40.add(Flatten())
663.cnn_model_bac_40.add(Dense(50, activation='relu'))
664.cnn_model_bac_40.add(Dense(1))
665.cnn_model_bac_40.compile(loss='mse',                optimizer='adam',
    metrics=['mse', 'mae', 'mape'])

666.cnn_model_bac_50 = Sequential()
667.cnn_model_bac_50.add(Conv1D(filters=64,                kernel_size=2,
    activation='relu', input_shape=(bac50_X_train.shape[1], 1)))
668.cnn_model_bac_50.add(MaxPooling1D(pool_size=2))
669.cnn_model_bac_50.add(Conv1D(filters=32,                kernel_size=1,
    activation='relu'))
670.cnn_model_bac_50.add(MaxPooling1D(pool_size=1))
671.cnn_model_bac_50.add(Conv1D(filters=64,                kernel_size=1,
    activation='relu'))
672.cnn_model_bac_50.add(MaxPooling1D(pool_size=1))
673.cnn_model_bac_50.add(Conv1D(filters=32,                kernel_size=1,
    activation='relu'))
674.cnn_model_bac_50.add(MaxPooling1D(pool_size=1))
675.cnn_model_bac_50.add(Conv1D(filters=64,                kernel_size=1,
    activation='relu'))
676.cnn_model_bac_50.add(MaxPooling1D(pool_size=1))
677.cnn_model_bac_50.add(Flatten())
678.cnn_model_bac_50.add(Dense(50, activation='relu'))
679.cnn_model_bac_50.add(Dense(1))
680.cnn_model_bac_50.compile(loss='mse',                optimizer='adam',
    metrics=['mse', 'mae', 'mape'])

```



```

681.cnn_model_bac_60 = Sequential()
682.cnn_model_bac_60.add(Conv1D(filters=64, kernel_size=2,
activation='relu', input_shape=(bac60_X_train.shape[1], 1)))
683.cnn_model_bac_60.add(MaxPooling1D(pool_size=2))
684.cnn_model_bac_60.add(Conv1D(filters=32, kernel_size=1,
activation='relu'))
685.cnn_model_bac_60.add(MaxPooling1D(pool_size=1))
686.cnn_model_bac_60.add(Conv1D(filters=64, kernel_size=1,
activation='relu'))
687.cnn_model_bac_60.add(MaxPooling1D(pool_size=1))
688.cnn_model_bac_60.add(Conv1D(filters=32, kernel_size=1,
activation='relu'))
689.cnn_model_bac_60.add(MaxPooling1D(pool_size=1))
690.cnn_model_bac_60.add(Conv1D(filters=64, kernel_size=1,
activation='relu'))
691.cnn_model_bac_60.add(MaxPooling1D(pool_size=1))
692.cnn_model_bac_60.add(Flatten())
693.cnn_model_bac_60.add(Dense(50, activation='relu'))
694.cnn_model_bac_60.add(Dense(1))
695.cnn_model_bac_60.compile(loss='mse', optimizer='adam',
metrics=['mse', 'mae', 'mape'])

696.cnn_model_bac_80 = Sequential()
697.cnn_model_bac_80.add(Conv1D(filters=64, kernel_size=2,
activation='relu', input_shape=(bac80_X_train.shape[1], 1)))
698.cnn_model_bac_80.add(MaxPooling1D(pool_size=2))
699.cnn_model_bac_80.add(Conv1D(filters=32, kernel_size=1,
activation='relu'))
700.cnn_model_bac_80.add(MaxPooling1D(pool_size=1))
701.cnn_model_bac_80.add(Conv1D(filters=64, kernel_size=1,
activation='relu'))
702.cnn_model_bac_80.add(MaxPooling1D(pool_size=1))
703.cnn_model_bac_80.add(Conv1D(filters=32, kernel_size=1,
activation='relu'))
704.cnn_model_bac_80.add(MaxPooling1D(pool_size=1))
705.cnn_model_bac_80.add(Conv1D(filters=64, kernel_size=1,
activation='relu'))
706.cnn_model_bac_80.add(MaxPooling1D(pool_size=1))
707.cnn_model_bac_80.add(Flatten())
708.cnn_model_bac_80.add(Dense(50, activation='relu'))
709.cnn_model_bac_80.add(Dense(1))
710.cnn_model_bac_80.compile(loss='mse', optimizer='adam',
metrics=['mse', 'mae', 'mape'])

```

BUILD CNN FOR DATASET HDB

```

711.cnn_model_hdb_20 = Sequential()
712.cnn_model_hdb_20.add(Conv1D(filters=64, kernel_size=2,
activation='relu', input_shape=(hdb20_X_train.shape[1], 1)))
713.cnn_model_hdb_20.add(MaxPooling1D(pool_size=2))
714.cnn_model_hdb_20.add(Conv1D(filters=32, kernel_size=1,
activation='relu'))
715.cnn_model_hdb_20.add(MaxPooling1D(pool_size=1))
716.cnn_model_hdb_20.add(Conv1D(filters=64, kernel_size=1,
activation='relu'))
717.cnn_model_hdb_20.add(MaxPooling1D(pool_size=1))
718.cnn_model_hdb_20.add(Conv1D(filters=32, kernel_size=1,
activation='relu'))
719.cnn_model_hdb_20.add(MaxPooling1D(pool_size=1))

```

```

720.cnn_model_hdb_20.add(Conv1D(filters=64,                                kernel_size=1,
    activation='relu'))
721.cnn_model_hdb_20.add(MaxPooling1D(pool_size=1))
722.cnn_model_hdb_20.add(Flatten())
723.cnn_model_hdb_20.add(Dense(50, activation='relu'))
724.cnn_model_hdb_20.add(Dense(1))
725.cnn_model_hdb_20.compile(loss='mse',                                optimizer='adam',
    metrics=['mse', 'mae', 'mape'])

726.cnn_model_hdb_40 = Sequential()
727.cnn_model_hdb_40.add(Conv1D(filters=64,                                kernel_size=2,
    activation='relu', input_shape=(hdb40_X_train.shape[1], 1)))
728.cnn_model_hdb_40.add(MaxPooling1D(pool_size=2))
729.cnn_model_hdb_40.add(Conv1D(filters=32,                                kernel_size=1,
    activation='relu'))
730.cnn_model_hdb_40.add(MaxPooling1D(pool_size=1))
731.cnn_model_hdb_40.add(Conv1D(filters=64,                                kernel_size=1,
    activation='relu'))
732.cnn_model_hdb_40.add(MaxPooling1D(pool_size=1))
733.cnn_model_hdb_40.add(Conv1D(filters=32,                                kernel_size=1,
    activation='relu'))
734.cnn_model_hdb_40.add(MaxPooling1D(pool_size=1))
735.cnn_model_hdb_40.add(Conv1D(filters=64,                                kernel_size=1,
    activation='relu'))
736.cnn_model_hdb_40.add(MaxPooling1D(pool_size=1))
737.cnn_model_hdb_40.add(Flatten())
738.cnn_model_hdb_40.add(Dense(50, activation='relu'))
739.cnn_model_hdb_40.add(Dense(1))
740.cnn_model_hdb_40.compile(loss='mse',                                optimizer='adam',
    metrics=['mse', 'mae', 'mape'])

741.cnn_model_hdb_50 = Sequential()
742.cnn_model_hdb_50.add(Conv1D(filters=64,                                kernel_size=2,
    activation='relu', input_shape=(hdb50_X_train.shape[1], 1)))
743.cnn_model_hdb_50.add(MaxPooling1D(pool_size=2))
744.cnn_model_hdb_50.add(Conv1D(filters=32,                                kernel_size=1,
    activation='relu'))
745.cnn_model_hdb_50.add(MaxPooling1D(pool_size=1))
746.cnn_model_hdb_50.add(Conv1D(filters=64,                                kernel_size=1,
    activation='relu'))
747.cnn_model_hdb_50.add(MaxPooling1D(pool_size=1))
748.cnn_model_hdb_50.add(Conv1D(filters=32,                                kernel_size=1,
    activation='relu'))
749.cnn_model_hdb_50.add(MaxPooling1D(pool_size=1))
750.cnn_model_hdb_50.add(Conv1D(filters=64,                                kernel_size=1,
    activation='relu'))
751.cnn_model_hdb_50.add(MaxPooling1D(pool_size=1))
752.cnn_model_hdb_50.add(Flatten())
753.cnn_model_hdb_50.add(Dense(50, activation='relu'))
754.cnn_model_hdb_50.add(Dense(1))
755.cnn_model_hdb_50.compile(loss='mse',                                optimizer='adam',
    metrics=['mse', 'mae', 'mape'])

756.cnn_model_hdb_60 = Sequential()
757.cnn_model_hdb_60.add(Conv1D(filters=64,                                kernel_size=2,
    activation='relu', input_shape=(hdb60_X_train.shape[1], 1)))
758.cnn_model_hdb_60.add(MaxPooling1D(pool_size=2))
759.cnn_model_hdb_60.add(Conv1D(filters=32,                                kernel_size=1,
    activation='relu'))

```

```

760.cnn_model_hdb_60.add(MaxPooling1D(pool_size=1))
761.cnn_model_hdb_60.add(Conv1D(filters=64, kernel_size=1,
activation='relu'))
762.cnn_model_hdb_60.add(MaxPooling1D(pool_size=1))
763.cnn_model_hdb_60.add(Conv1D(filters=32, kernel_size=1,
activation='relu'))
764.cnn_model_hdb_60.add(MaxPooling1D(pool_size=1))
765.cnn_model_hdb_60.add(Conv1D(filters=64, kernel_size=1,
activation='relu'))
766.cnn_model_hdb_60.add(MaxPooling1D(pool_size=1))
767.cnn_model_hdb_60.add(Flatten())
768.cnn_model_hdb_60.add(Dense(50, activation='relu'))
769.cnn_model_hdb_60.add(Dense(1))
770.cnn_model_hdb_60.compile(loss='mse', optimizer='adam',
metrics=['mse', 'mae', 'mape'])

771.cnn_model_hdb_80 = Sequential()
772.cnn_model_hdb_80.add(Conv1D(filters=64, kernel_size=2,
activation='relu', input_shape=(hdb80_X_train.shape[1], 1)))
773.cnn_model_hdb_80.add(MaxPooling1D(pool_size=2))
774.cnn_model_hdb_80.add(Conv1D(filters=32, kernel_size=1,
activation='relu'))
775.cnn_model_hdb_80.add(MaxPooling1D(pool_size=1))
776.cnn_model_hdb_80.add(Conv1D(filters=64, kernel_size=1,
activation='relu'))
777.cnn_model_hdb_80.add(MaxPooling1D(pool_size=1))
778.cnn_model_hdb_80.add(Conv1D(filters=32, kernel_size=1,
activation='relu'))
779.cnn_model_hdb_80.add(MaxPooling1D(pool_size=1))
780.cnn_model_hdb_80.add(Conv1D(filters=64, kernel_size=1,
activation='relu'))
781.cnn_model_hdb_80.add(MaxPooling1D(pool_size=1))
782.cnn_model_hdb_80.add(Flatten())
783.cnn_model_hdb_80.add(Dense(50, activation='relu'))
784.cnn_model_hdb_80.add(Dense(1))
785.cnn_model_hdb_80.compile(loss='mse', optimizer='adam',
metrics=['mse', 'mae', 'mape'])

```

BUILD CNN FOR DATASET RY

```

786.cnn_model_ry_20 = Sequential()
787.cnn_model_ry_20.add(Conv1D(filters=64, kernel_size=2,
activation='relu', input_shape=(ry20_X_train.shape[1], 1)))
788.cnn_model_ry_20.add(MaxPooling1D(pool_size=2))
789.cnn_model_ry_20.add(Conv1D(filters=32, kernel_size=1,
activation='relu'))
790.cnn_model_ry_20.add(MaxPooling1D(pool_size=1))
791.cnn_model_ry_20.add(Conv1D(filters=64, kernel_size=1,
activation='relu'))
792.cnn_model_ry_20.add(MaxPooling1D(pool_size=1))
793.cnn_model_ry_20.add(Conv1D(filters=32, kernel_size=1,
activation='relu'))
794.cnn_model_ry_20.add(MaxPooling1D(pool_size=1))
795.cnn_model_ry_20.add(Conv1D(filters=64, kernel_size=1,
activation='relu'))
796.cnn_model_ry_20.add(MaxPooling1D(pool_size=1))
797.cnn_model_ry_20.add(Flatten())
798.cnn_model_ry_20.add(Dense(50, activation='relu'))
799.cnn_model_ry_20.add(Dense(1))

```

```

800.cnn_model_ry_20.compile(loss='mse', optimizer='adam',
    metrics=['mse', 'mae', 'mape'])

801.cnn_model_ry_40 = Sequential()
802.cnn_model_ry_40.add(Conv1D(filters=64, kernel_size=2,
    activation='relu', input_shape=(ry40_X_train.shape[1], 1)))
803.cnn_model_ry_40.add(MaxPooling1D(pool_size=2))
804.cnn_model_ry_40.add(Conv1D(filters=32, kernel_size=1,
    activation='relu'))
805.cnn_model_ry_40.add(MaxPooling1D(pool_size=1))
806.cnn_model_ry_40.add(Conv1D(filters=64, kernel_size=1,
    activation='relu'))
807.cnn_model_ry_40.add(MaxPooling1D(pool_size=1))
808.cnn_model_ry_40.add(Conv1D(filters=32, kernel_size=1,
    activation='relu'))
809.cnn_model_ry_40.add(MaxPooling1D(pool_size=1))
810.cnn_model_ry_40.add(Conv1D(filters=64, kernel_size=1,
    activation='relu'))
811.cnn_model_ry_40.add(MaxPooling1D(pool_size=1))
812.cnn_model_ry_40.add(Flatten())
813.cnn_model_ry_40.add(Dense(50, activation='relu'))
814.cnn_model_ry_40.add(Dense(1))
815.cnn_model_ry_40.compile(loss='mse', optimizer='adam',
    metrics=['mse', 'mae', 'mape'])

816.cnn_model_ry_50 = Sequential()
817.cnn_model_ry_50.add(Conv1D(filters=64, kernel_size=2,
    activation='relu', input_shape=(ry50_X_train.shape[1], 1)))
818.cnn_model_ry_50.add(MaxPooling1D(pool_size=2))
819.cnn_model_ry_50.add(Conv1D(filters=32, kernel_size=1,
    activation='relu'))
820.cnn_model_ry_50.add(MaxPooling1D(pool_size=1))
821.cnn_model_ry_50.add(Conv1D(filters=64, kernel_size=1,
    activation='relu'))
822.cnn_model_ry_50.add(MaxPooling1D(pool_size=1))
823.cnn_model_ry_50.add(Conv1D(filters=32, kernel_size=1,
    activation='relu'))
824.cnn_model_ry_50.add(MaxPooling1D(pool_size=1))
825.cnn_model_ry_50.add(Conv1D(filters=64, kernel_size=1,
    activation='relu'))
826.cnn_model_ry_50.add(MaxPooling1D(pool_size=1))
827.cnn_model_ry_50.add(Flatten())
828.cnn_model_ry_50.add(Dense(50, activation='relu'))
829.cnn_model_ry_50.add(Dense(1))
830.cnn_model_ry_50.compile(loss='mse', optimizer='adam',
    metrics=['mse', 'mae', 'mape'])

831.cnn_model_ry_60 = Sequential()
832.cnn_model_ry_60.add(Conv1D(filters=64, kernel_size=2,
    activation='relu', input_shape=(ry60_X_train.shape[1], 1)))
833.cnn_model_ry_60.add(MaxPooling1D(pool_size=2))
834.cnn_model_ry_60.add(Conv1D(filters=32, kernel_size=1,
    activation='relu'))
835.cnn_model_ry_60.add(MaxPooling1D(pool_size=1))
836.cnn_model_ry_60.add(Conv1D(filters=64, kernel_size=1,
    activation='relu'))
837.cnn_model_ry_60.add(MaxPooling1D(pool_size=1))
838.cnn_model_ry_60.add(Conv1D(filters=32, kernel_size=1,
    activation='relu'))

```

```

839.cnn_model_ry_60.add(MaxPooling1D(pool_size=1))
840.cnn_model_ry_60.add(Conv1D(filters=64,                                kernel_size=1,
    activation='relu'))
841.cnn_model_ry_60.add(MaxPooling1D(pool_size=1))
842.cnn_model_ry_60.add(Flatten())
843.cnn_model_ry_60.add(Dense(50, activation='relu'))
844.cnn_model_ry_60.add(Dense(1))
845.cnn_model_ry_60.compile(loss='mse',                                optimizer='adam',
    metrics=['mse', 'mae', 'mape'])

846.cnn_model_ry_80 = Sequential()
847.cnn_model_ry_80.add(Conv1D(filters=64,                                kernel_size=2,
    activation='relu', input_shape=(ry80_X_train.shape[1], 1)))
848.cnn_model_ry_80.add(MaxPooling1D(pool_size=2))
849.cnn_model_ry_80.add(Conv1D(filters=32,                                kernel_size=1,
    activation='relu'))
850.cnn_model_ry_80.add(MaxPooling1D(pool_size=1))
851.cnn_model_ry_80.add(Conv1D(filters=64,                                kernel_size=1,
    activation='relu'))
852.cnn_model_ry_80.add(MaxPooling1D(pool_size=1))
853.cnn_model_ry_80.add(Conv1D(filters=32,                                kernel_size=1,
    activation='relu'))
854.cnn_model_ry_80.add(MaxPooling1D(pool_size=1))
855.cnn_model_ry_80.add(Conv1D(filters=64,                                kernel_size=1,
    activation='relu'))
856.cnn_model_ry_80.add(MaxPooling1D(pool_size=1))
857.cnn_model_ry_80.add(Flatten())
858.cnn_model_ry_80.add(Dense(50, activation='relu'))
859.cnn_model_ry_80.add(Dense(1))
860.cnn_model_ry_80.compile(loss='mse',                                optimizer='adam',
    metrics=['mse', 'mae', 'mape'])

```

EARLY STOPPING

```

861.earlyStop = EarlyStopping(monitor='loss', mode='min', verbose=1,
    patience=50)

```

EVALUATION TESTING 20%

```

862.history_bac_20=model_bac_20.fit(bac20_X_train,bac20_y_train,epochs=5
    00,batch_size=64,validation_data=(bac20_X_test,
    bac20_y_test),callbacks=[earlyStop])
863.batch_size = 1
864.trainPredict_bac20 = model_bac_20.predict(bac20_X_train,
    batch_size=batch_size)
865.model_bac_20.reset_states()
866.testPredict_bac20 = model_bac_20.predict(bac20_X_test,
    batch_size=batch_size)
867.bac20_trainPredict2=sc.fit_transform(trainPredict_bac20)
868.bac20_trainPredict=sc.inverse_transform(bac20_trainPredict2)
869.bac20_trainY2 = sc.fit_transform([bac20_y_train])
870.bac20_trainY = sc.inverse_transform(bac20_trainY2)
871.bac20_testPredict2 = sc.fit_transform(testPredict_bac20)
872.bac20_testPredict = sc.inverse_transform(bac20_testPredict2)
873.bac20_testY2 = sc.fit_transform([bac20_y_test])
874.bac20_testY = sc.inverse_transform(bac20_testY2)

875.cnn_history_bac_20=cnn_model_bac_20.fit(bac20_X_train,bac20_y_train,
    epochs=500,batch_size=64,validation_data=(bac20_X_test,
    bac20_y_test),callbacks=[earlyStop])
876.batch_size = 1

```

```

877.cnn_trainPredict_bac20 = cnn_model_bac_20.predict(bac20_X_train,
    batch_size=batch_size)
878.cnn_model_bac_20.reset_states()
879.cnn_testPredict_bac20 = cnn_model_bac_20.predict(bac20_X_test,
    batch_size=batch_size)
880.cnn_bac20_trainPredict2=sc.fit_transform(cnn_trainPredict_bac20)
881.cnn_bac20_trainPredict=sc.inverse_transform(cnn_bac20_trainPredict2)
882.cnn_bac20_trainY2 = sc.fit_transform([bac20_y_train])
883.cnn_bac20_trainY = sc.inverse_transform(cnn_bac20_trainY2)
884.cnn_bac20_testPredict2 = sc.fit_transform(cnn_testPredict_bac20)
885.cnn_bac20_testPredict = sc.inverse_transform(cnn_bac20_testPredict2)
886.cnn_bac20_testY2 = sc.fit_transform([bac20_y_test])
887.cnn_bac20_testY = sc.inverse_transform(cnn_bac20_testY2)

888.lst = [
889.     ['%.3f' % mae(bac20_y_test, bac20_linear_reg_pred), '%.3f' %
    mae(bac20_y_test, bac20_lasso_reg_pred), '%.3f' % mae(bac20_y_test,
    bac20_gaussian_pred), '%.3f' % mae(bac20_y_test,
    bac20_testPredict[:,0]), '%.3f' % mae(bac20_y_test,
    cnn_bac20_testPredict[:,0])],
890.     ['%.3f' % rmse(bac20_y_test, bac20_linear_reg_pred), '%.3f'
    % rmse(bac20_y_test, bac20_lasso_reg_pred), '%.3f' %
    rmse(bac20_y_test, bac20_gaussian_pred), '%.3f' % rmse(bac20_y_test,
    bac20_testPredict[:,0]), '%.3f' % rmse(bac20_y_test,
    cnn_bac20_testPredict[:,0])],
891.     ['%.3f' % mape(bac20_y_test, bac20_linear_reg_pred), '%.3f'
    % mape(bac20_y_test, bac20_lasso_reg_pred), '%.3f' %
    mape(bac20_y_test, bac20_gaussian_pred), '%.3f' % mape(bac20_y_test,
    bac20_testPredict[:,0]), '%.3f' % mape(bac20_y_test,
    cnn_bac20_testPredict[:,0])]
892. ]
893.df = pd.DataFrame(lst,
894.     columns =['Linear Regression', 'Lasso Regression',
    'LSSVM', 'LSTM', 'CNN'],
895.     index=['MAE', 'RMSE', 'MAPE'])
896.df

897.history_hdb_20=model_hdb_20.fit(hdb20_X_train,hdb20_y_train,epochs=5
    00,batch_size=64,validation_data=(hdb20_X_test,
    hdb20_y_test),callbacks=[earlyStop])
898.batch_size = 1
899.trainPredict_hdb20 = model_hdb_20.predict(hdb20_X_train,
    batch_size=batch_size)
900.model_hdb_20.reset_states()
901.testPredict_hdb20 = model_hdb_20.predict(hdb20_X_test,
    batch_size=batch_size)
902.hdb20_trainPredict2=sc.fit_transform(trainPredict_hdb20)
903.hdb20_trainPredict=sc.inverse_transform(hdb20_trainPredict2)
904.hdb20_trainY2 = sc.fit_transform([hdb20_y_train])
905.hdb20_trainY = sc.inverse_transform(hdb20_trainY2)
906.hdb20_testPredict2 = sc.fit_transform(testPredict_hdb20)
907.hdb20_testPredict = sc.inverse_transform(hdb20_testPredict2)
908.hdb20_testY2 = sc.fit_transform([hdb20_y_test])
909.hdb20_testY = sc.inverse_transform(hdb20_testY2)

910.cnn_history_hdb_20=cnn_model_hdb_20.fit(hdb20_X_train,hdb20_y_train,
    epochs=500,batch_size=64,validation_data=(hdb20_X_test,
    hdb20_y_test),callbacks=[earlyStop])

```

```

911.batch_size = 1
912.cnn_trainPredict_hdb20 = cnn_model_hdb_20.predict(hdb20_X_train,
    batch_size=batch_size)
913.cnn_model_hdb_20.reset_states()
914.cnn_testPredict_hdb20 = cnn_model_hdb_20.predict(hdb20_X_test,
    batch_size=batch_size)
915.cnn_hdb20_trainPredict2=sc.fit_transform(cnn_trainPredict_hdb20)
916.cnn_hdb20_trainPredict=sc.inverse_transform(cnn_hdb20_trainPredict2)
917.cnn_hdb20_trainY2 = sc.fit_transform([hdb20_y_train])
918.cnn_hdb20_trainY = sc.inverse_transform(cnn_hdb20_trainY2)
919.cnn_hdb20_testPredict2 = sc.fit_transform(cnn_testPredict_hdb20)
920.cnn_hdb20_testPredict = sc.inverse_transform(cnn_hdb20_testPredict2)
921.cnn_hdb20_testY2 = sc.fit_transform([hdb20_y_test])
922.cnn_hdb20_testY = sc.inverse_transform(cnn_hdb20_testY2)

923.lst = [
924.     ['%.3f' % mae(hdb20_y_test, hdb20_linear_reg_pred), '%.3f' %
    mae(hdb20_y_test, hdb20_lasso_reg_pred), '%.3f' % mae(hdb20_y_test,
    hdb20_gaussian_pred), '%.3f' % mae(hdb20_y_test,
    hdb20_testPredict[:,0]), '%.3f' % mae(hdb20_y_test,
    cnn_hdb20_testPredict[:,0])],
925.     ['%.3f' % rmse(hdb20_y_test, hdb20_linear_reg_pred), '%.3f'
    % rmse(hdb20_y_test, hdb20_lasso_reg_pred), '%.3f' %
    rmse(hdb20_y_test, hdb20_gaussian_pred), '%.3f' % rmse(hdb20_y_test,
    hdb20_testPredict[:,0]), '%.3f' % rmse(hdb20_y_test,
    cnn_hdb20_testPredict[:,0])],
926.     ['%.3f' % mape(hdb20_y_test, hdb20_linear_reg_pred), '%.3f'
    % mape(hdb20_y_test, hdb20_lasso_reg_pred), '%.3f' %
    mape(hdb20_y_test, hdb20_gaussian_pred), '%.3f' % mape(hdb20_y_test,
    hdb20_testPredict[:,0]), '%.3f' % mape(hdb20_y_test,
    cnn_hdb20_testPredict[:,0])]
927. ]
928.df = pd.DataFrame(lst,
929.     columns=['Linear Regression', 'Lasso Regression',
    'LSSVM', 'LSTM', 'CNN'],
930.     index=['MAE', 'RMSE', 'MAPE'])
931.df

932.history_ry_20=model_ry_20.fit(ry20_X_train,ry20_y_train,epochs=500,b
    atch_size=64,validation_data=(ry20_X_test,
    ry20_y_test),callbacks=[earlyStop])
933.batch_size = 1
934.trainPredict_ry20 = model_ry_20.predict(ry20_X_train,
    batch_size=batch_size)
935.model_ry_20.reset_states()
936.testPredict_ry20 = model_ry_20.predict(ry20_X_test,
    batch_size=batch_size)
937.ry20_trainPredict2=sc.fit_transform(trainPredict_ry20)
938.ry20_trainPredict=sc.inverse_transform(ry20_trainPredict2)
939.ry20_trainY2 = sc.fit_transform([ry20_y_train])
940.ry20_trainY = sc.inverse_transform(ry20_trainY2)
941.ry20_testPredict2 = sc.fit_transform(testPredict_ry20)
942.ry20_testPredict = sc.inverse_transform(ry20_testPredict2)
943.ry20_testY2 = sc.fit_transform([ry20_y_test])
944.ry20_testY = sc.inverse_transform(ry20_testY2)

```

```

945.cnn_history_ry_20=cnn_model_ry_20.fit(ry20_X_train,ry20_y_train,epoch
    hs=500,batch_size=64,validation_data=(ry20_X_test,
    ry20_y_test),callbacks=[earlyStop])
946.batch_size = 1
947.cnn_trainPredict_ry20      =      cnn_model_ry_20.predict(ry20_X_train,
    batch_size=batch_size)
948.cnn_model_ry_20.reset_states()
949.cnn_testPredict_ry20      =      cnn_model_ry_20.predict(ry20_X_test,
    batch_size=batch_size)
950.cnn_ry20_trainPredict2=sc.fit_transform(cnn_trainPredict_ry20)
951.cnn_ry20_trainPredict=sc.inverse_transform(cnn_ry20_trainPredict2)
952.cnn_ry20_trainY2 = sc.fit_transform([ry20_y_train])
953.cnn_ry20_trainY = sc.inverse_transform(cnn_ry20_trainY2)
954.cnn_ry20_testPredict2 = sc.fit_transform(cnn_testPredict_ry20)
955.cnn_ry20_testPredict = sc.inverse_transform(cnn_ry20_testPredict2)
956.cnn_ry20_testY2 = sc.fit_transform([ry20_y_test])
957.cnn_ry20_testY = sc.inverse_transform(cnn_ry20_testY2)

958.lst = [
959.      ['%.3f' % mae(ry20_y_test, ry20_linear_reg_pred), '%.3f' %
    mae(ry20_y_test, ry20_lasso_reg_pred), '%.3f' % mae(ry20_y_test,
    ry20_gaussian_pred), '%.3f' % mae(ry20_y_test, ry20_testPredict[:,0]),
    '%.3f' % mae(ry20_y_test, cnn_ry20_testPredict[:,0])],
960.      ['%.3f' % rmse(ry20_y_test, ry20_linear_reg_pred), '%.3f' %
    rmse(ry20_y_test, ry20_lasso_reg_pred), '%.3f' % rmse(ry20_y_test,
    ry20_gaussian_pred), '%.3f' % rmse(ry20_y_test,
    ry20_testPredict[:,0]), '%.3f' % rmse(ry20_y_test,
    cnn_ry20_testPredict[:,0])],
961.      ['%.3f' % mape(ry20_y_test, ry20_linear_reg_pred), '%.3f' %
    mape(ry20_y_test, ry20_lasso_reg_pred), '%.3f' % mape(ry20_y_test,
    ry20_gaussian_pred), '%.3f' % mape(ry20_y_test,
    ry20_testPredict[:,0]), '%.3f' % mape(ry20_y_test,
    cnn_ry20_testPredict[:,0])]
962.      ]
963.df = pd.DataFrame(lst,
964.      columns =['Linear Regression', 'Lasso Regression',
    'LSSVM', 'LSTM', 'CNN'],
965.      index=['MAE', 'RMSE', 'MAPE'])
966.df

```

EVALUATION TESTING 40%

```

967.history_bac_40=model_bac_40.fit(bac40_X_train,bac40_y_train,epochs=5
    00,batch_size=64,validation_data=(bac40_X_test,
    bac40_y_test),callbacks=[earlyStop])
968.batch_size = 1
969.trainPredict_bac40      =      model_bac_40.predict(bac40_X_train,
    batch_size=batch_size)
970.model_bac_40.reset_states()
971.testPredict_bac40      =      model_bac_40.predict(bac40_X_test,
    batch_size=batch_size)
972.bac40_trainPredict2=sc.fit_transform(trainPredict_bac40)
973.bac40_trainPredict=sc.inverse_transform(bac40_trainPredict2)
974.
975.bac40_trainY2 = sc.fit_transform([bac40_y_train])
976.bac40_trainY = sc.inverse_transform(bac40_trainY2)
977.bac40_testPredict2 = sc.fit_transform(testPredict_bac40)
978.bac40_testPredict = sc.inverse_transform(bac40_testPredict2)
979.bac40_testY2 = sc.fit_transform([bac40_y_test])
980.bac40_testY = sc.inverse_transform(bac40_testY2)

```



```

981.cnn_history_bac_40=cnn_model_bac_40.fit(bac40_X_train,bac40_y_train,
      epochs=500,batch_size=64,validation_data=(bac40_X_test,
      bac40_y_test),callbacks=[earlyStop])
982.batch_size = 1
983.cnn_trainPredict_bac40 = cnn_model_bac_40.predict(bac40_X_train,
      batch_size=batch_size)
984.cnn_model_bac_40.reset_states()
985.cnn_testPredict_bac40 = cnn_model_bac_40.predict(bac40_X_test,
      batch_size=batch_size)
986.cnn_bac40_trainPredict2=sc.fit_transform(cnn_trainPredict_bac40)
987.cnn_bac40_trainPredict=sc.inverse_transform(cnn_bac40_trainPredict2)
988.cnn_bac40_trainY2 = sc.fit_transform([bac40_y_train])
989.cnn_bac40_trainY = sc.inverse_transform(cnn_bac40_trainY2)
990.cnn_bac40_testPredict2 = sc.fit_transform(cnn_testPredict_bac40)
991.cnn_bac40_testPredict = sc.inverse_transform(cnn_bac40_testPredict2)
992.cnn_bac40_testY2 = sc.fit_transform([bac40_y_test])
993.cnn_bac40_testY = sc.inverse_transform(cnn_bac40_testY2)

994.lst = [
995.      ['%.3f' % mae(bac40_y_test, bac40_linear_reg_pred), '%.3f' %
      mae(bac40_y_test, bac40_lasso_reg_pred), '%.3f' % mae(bac40_y_test,
      bac40_gaussian_pred), '%.3f' % mae(bac40_y_test,
      bac40_testPredict[:,0]), '%.3f' % mae(bac40_y_test,
      cnn_bac40_testPredict[:,0])],
996.      ['%.3f' % rmse(bac40_y_test, bac40_linear_reg_pred), '%.3f'
      % rmse(bac40_y_test, bac40_lasso_reg_pred), '%.3f' %
      rmse(bac40_y_test, bac40_gaussian_pred), '%.3f' % rmse(bac40_y_test,
      bac40_testPredict[:,0]), '%.3f' % rmse(bac40_y_test,
      cnn_bac40_testPredict[:,0])],
997.      ['%.3f' % mape(bac40_y_test, bac40_linear_reg_pred), '%.3f'
      % mape(bac40_y_test, bac40_lasso_reg_pred), '%.3f' %
      mape(bac40_y_test, bac40_gaussian_pred), '%.3f' % mape(bac40_y_test,
      bac40_testPredict[:,0]), '%.3f' % mape(bac40_y_test,
      cnn_bac40_testPredict[:,0])]
998.      ]
999.df = pd.DataFrame(lst,
1000.      columns =['Linear Regression', 'Lasso
      Regression', 'LSSVM', 'LSTM', 'CNN'],
1001.      index=['MAE', 'RMSE', 'MAPE'])
1002.      df

1003.      history_hdb_40=model_hdb_40.fit(hdb40_X_train,hdb40_y_train,epo
      chs=500,batch_size=64,validation_data=(hdb40_X_test,
      hdb40_y_test),callbacks=[earlyStop])
1004.      batch_size = 1
1005.      trainPredict_bac40 = model_hdb_40.predict(hdb40_X_train,
      batch_size=batch_size)
1006.      model_hdb_40.reset_states()
1007.      testPredict_bac40 = model_hdb_40.predict(hdb40_X_test,
      batch_size=batch_size)
1008.      bac40_trainPredict2=sc.fit_transform(trainPredict_bac40)
1009.      hdb40_trainPredict=sc.inverse_transform(bac40_trainPredict2)
1010.      bac40_trainY2 = sc.fit_transform([hdb40_y_train])
1011.      hdb40_trainY = sc.inverse_transform(bac40_trainY2)
1012.      bac40_testPredict2 = sc.fit_transform(testPredict_bac40)
1013.      hdb40_testPredict = sc.inverse_transform(bac40_testPredict2)
1014.      bac40_testY2 = sc.fit_transform([hdb40_y_test])

```

```

1015.     hdb40_testY = sc.inverse_transform(bac40_testY2)

1016.     cnn_history_hdb_40=cnn_model_hdb_40.fit(hdb40_X_train,hdb40_y_train,epochs=500,batch_size=64,validation_data=(hdb40_X_test,hdb40_y_test),callbacks=[earlyStop])
1017.     batch_size = 1
1018.     cnn_trainPredict_hdb40 = cnn_model_hdb_40.predict(hdb40_X_train, batch_size=batch_size)
1019.     cnn_model_hdb_40.reset_states()
1020.     cnn_testPredict_hdb40 = cnn_model_hdb_40.predict(hdb40_X_test, batch_size=batch_size)
1021.     cnn_hdb40_trainPredict2=sc.fit_transform(cnn_trainPredict_hdb40)
1022.     cnn_hdb40_trainPredict=sc.inverse_transform(cnn_hdb40_trainPredict2)
1023.     cnn_hdb40_trainY2 = sc.fit_transform([hdb40_y_train])
1024.     cnn_hdb40_trainY = sc.inverse_transform(cnn_hdb40_trainY2)
1025.     cnn_hdb40_testPredict2 = sc.fit_transform(cnn_testPredict_hdb40)
1026.     cnn_hdb40_testPredict = sc.inverse_transform(cnn_hdb40_testPredict2)
1027.     cnn_hdb40_testY2 = sc.fit_transform([hdb40_y_test])
1028.     cnn_hdb40_testY = sc.inverse_transform(cnn_hdb40_testY2)

1029.     lst = [
1030.         ['%.3f' % mae(hdb40_y_test, hdb40_linear_reg_pred), '%.3f' % mae(hdb40_y_test, hdb40_lasso_reg_pred), '%.3f' % mae(hdb40_y_test, hdb40_gaussian_pred), '%.3f' % mae(hdb40_y_test, hdb40_testPredict[:,0]), '%.3f' % mae(hdb40_y_test, cnn_hdb40_testPredict[:,0])],
1031.         ['%.3f' % rmse(hdb40_y_test, hdb40_linear_reg_pred), '%.3f' % rmse(hdb40_y_test, hdb40_lasso_reg_pred), '%.3f' % rmse(hdb40_y_test, hdb40_gaussian_pred), '%.3f' % rmse(hdb40_y_test, hdb40_testPredict[:,0]), '%.3f' % rmse(hdb40_y_test, cnn_hdb40_testPredict[:,0])],
1032.         ['%.3f' % mape(hdb40_y_test, hdb40_linear_reg_pred), '%.3f' % mape(hdb40_y_test, hdb40_lasso_reg_pred), '%.3f' % mape(hdb40_y_test, hdb40_gaussian_pred), '%.3f' % mape(hdb40_y_test, hdb40_testPredict[:,0]), '%.3f' % mape(hdb40_y_test, cnn_hdb40_testPredict[:,0])],
1033.     ]
1034.     df = pd.DataFrame(lst,
1035.         columns = ['Linear Regression', 'Lasso Regression', 'LSSVM', 'LSTM', 'CNN'],
1036.         index=['MAE', 'RMSE', 'MAPE'])
1037.     df

1038.     history_ry_40=model_ry_40.fit(ry40_X_train,ry40_y_train,epochs=500,batch_size=64,validation_data=(ry40_X_test,ry40_y_test),callbacks=[earlyStop])
1039.     batch_size = 1
1040.     trainPredict_ry40 = model_ry_40.predict(ry40_X_train, batch_size=batch_size)
1041.     model_ry_40.reset_states()
1042.     testPredict_ry40 = model_ry_40.predict(ry40_X_test, batch_size=batch_size)
1043.     ry40_trainPredict2=sc.fit_transform(trainPredict_ry40)
1044.     ry40_trainPredict=sc.inverse_transform(ry40_trainPredict2)

```

```

1045. ry40_trainY2 = sc.fit_transform([ry40_y_train])
1046. ry40_trainY = sc.inverse_transform(ry40_trainY2)
1047. ry40_testPredict2 = sc.fit_transform(testPredict_ry40)
1048. ry40_testPredict = sc.inverse_transform(ry40_testPredict2)
1049. ry40_testY2 = sc.fit_transform([ry40_y_test])
1050. ry40_testY = sc.inverse_transform(ry40_testY2)

1051. cnn_history_ry_40=cnn_model_ry_40.fit(ry40_X_train,ry40_y_train
,epochs=500,batch_size=64,validation_data=(ry40_X_test,
ry40_y_test),callbacks=[earlyStop])
1052. batch_size = 1
1053. cnn_trainPredict_ry40 = cnn_model_ry_40.predict(ry40_X_train,
batch_size=batch_size)
1054. cnn_model_ry_40.reset_states()
1055. cnn_testPredict_ry40 = cnn_model_ry_40.predict(ry40_X_test,
batch_size=batch_size)
1056. cnn_ry40_trainPredict2=sc.fit_transform(cnn_trainPredict_ry40)
1057. cnn_ry40_trainPredict=sc.inverse_transform(cnn_ry40_trainPredic
t2)
1058. cnn_ry40_trainY2 = sc.fit_transform([ry40_y_train])
1059. cnn_ry40_trainY = sc.inverse_transform(cnn_ry40_trainY2)
1060. cnn_ry40_testPredict2 = sc.fit_transform(cnn_testPredict_ry40)
1061. cnn_ry40_testPredict =
sc.inverse_transform(cnn_ry40_testPredict2)
1062. cnn_ry40_testY2 = sc.fit_transform([ry40_y_test])
1063. cnn_ry40_testY = sc.inverse_transform(cnn_ry40_testY2)

1064. lst = [
1065.     ['%.3f' % mae(ry40_y_test, ry40_linear_reg_pred), '%.3f'
% mae(ry40_y_test, ry40_lasso_reg_pred), '%.3f' % mae(ry40_y_test,
ry40_gaussian_pred), '%.3f' % mae(ry40_y_test, ry40_testPredict[:,0]),
%.3f' % mae(ry40_y_test, cnn_ry40_testPredict[:,0])],
1066.     ['%.3f' % rmse(ry40_y_test, ry40_linear_reg_pred),
%.3f' % rmse(ry40_y_test, ry40_lasso_reg_pred), '%.3f' %
rmse(ry40_y_test, ry40_gaussian_pred), '%.3f' % rmse(ry40_y_test,
ry40_testPredict[:,0]), '%.3f' % rmse(ry40_y_test,
cnn_ry40_testPredict[:,0])],
1067.     ['%.3f' % mape(ry40_y_test, ry40_linear_reg_pred),
%.3f' % mape(ry40_y_test, ry40_lasso_reg_pred), '%.3f' %
mape(ry40_y_test, ry40_gaussian_pred), '%.3f' % mape(ry40_y_test,
ry40_testPredict[:,0]), '%.3f' % mape(ry40_y_test,
cnn_ry40_testPredict[:,0])]
1068.     ]
1069. df = pd.DataFrame(lst,
1070.     columns =['Linear Regression', 'Lasso
Regression', 'LSSVM', 'LSTM', 'CNN'],
1071.     index=['MAE', 'RMSE', 'MAPE'])
1072. df

```

EVALUATION TESTING 50%

```

1073. history_bac_50=model_bac_50.fit(bac50_X_train,bac50_y_train,epo
chs=500,batch_size=64,validation_data=(bac50_X_test,
bac50_y_test),callbacks=[earlyStop])
1074. batch_size = 1
1075. trainPredict_bac50 = model_bac_50.predict(bac50_X_train,
batch_size=batch_size)
1076. model_bac_50.reset_states()
1077. testPredict_bac50 = model_bac_50.predict(bac50_X_test,
batch_size=batch_size)

```

```

1078. bac50_trainPredict2=sc.fit_transform(trainPredict_bac50)
1079. bac50_trainPredict=sc.inverse_transform(bac50_trainPredict2)
1080. bac50_trainY2 = sc.fit_transform([bac50_y_train])
1081. bac50_trainY = sc.inverse_transform(bac50_trainY2)
1082. bac50_testPredict2 = sc.fit_transform(testPredict_bac50)
1083. bac50_testPredict = sc.inverse_transform(bac50_testPredict2)
1084. bac50_testY2 = sc.fit_transform([bac50_y_test])
1085. bac50_testY = sc.inverse_transform(bac50_testY2)

1086. cnn_history_bac_50=cnn_model_bac_50.fit(bac50_X_train,bac50_y_train,epochs=500,batch_size=64,validation_data=(bac50_X_test,bac50_y_test),callbacks=[earlyStop])
1087. batch_size = 1
1088. cnn_trainPredict_bac50 = cnn_model_bac_50.predict(bac50_X_train, batch_size=batch_size)
1089. cnn_model_bac_50.reset_states()
1090. cnn_testPredict_bac50 = cnn_model_bac_50.predict(bac50_X_test, batch_size=batch_size)
1091. cnn_bac50_trainPredict2=sc.fit_transform(cnn_trainPredict_bac50)
1092. cnn_bac50_trainPredict=sc.inverse_transform(cnn_bac50_trainPredict2)
1093. cnn_bac50_trainY2 = sc.fit_transform([bac50_y_train])
1094. cnn_bac50_trainY = sc.inverse_transform(cnn_bac50_trainY2)
1095. cnn_bac50_testPredict2 = sc.fit_transform(cnn_testPredict_bac50)
1096. cnn_bac50_testPredict = sc.inverse_transform(cnn_bac50_testPredict2)
1097. cnn_bac50_testY2 = sc.fit_transform([bac50_y_test])
1098. cnn_bac50_testY = sc.inverse_transform(cnn_bac50_testY2)

1099. lst = [
1100.     ['%.3f' % mae(bac50_y_test, bac50_linear_reg_pred),
%.3f' % mae(bac50_y_test, bac50_lasso_reg_pred), '%.3f' % mae(bac50_y_test, bac50_gaussian_pred), '%.3f' % mae(bac50_y_test, bac50_testPredict[:,0]), '%.3f' % mae(bac50_y_test, cnn_bac50_testPredict[:,0])],
1101.     ['%.3f' % rmse(bac50_y_test, bac50_linear_reg_pred),
%.3f' % rmse(bac50_y_test, bac50_lasso_reg_pred), '%.3f' % rmse(bac50_y_test, bac50_gaussian_pred), '%.3f' % rmse(bac50_y_test, bac50_testPredict[:,0]), '%.3f' % rmse(bac50_y_test, cnn_bac50_testPredict[:,0])],
1102.     ['%.3f' % mape(bac50_y_test, bac50_linear_reg_pred),
%.3f' % mape(bac50_y_test, bac50_lasso_reg_pred), '%.3f' % mape(bac50_y_test, bac50_gaussian_pred), '%.3f' % mape(bac50_y_test, bac50_testPredict[:,0]), '%.3f' % mape(bac50_y_test, cnn_bac50_testPredict[:,0])]
1103. ]
1104. df = pd.DataFrame(lst,
1105.     columns = ['Linear Regression', 'Lasso Regression', 'LSSVM', 'LSTM', 'CNN'],
1106.     index=['MAE', 'RMSE', 'MAPE'])
1107. df

1108. history_hdb_50=model_hdb_50.fit(hdb50_X_train,hdb50_y_train,epochs=500,batch_size=64,validation_data=(hdb50_X_test,hdb50_y_test),callbacks=[earlyStop])
1109. batch_size = 1

```

```

1110.     trainPredict_hdb50      =     model_hdb_50.predict(hdb50_X_train,
        batch_size=batch_size)
1111.     model_hdb_50.reset_states()
1112.     testPredict_hdb50       =     model_hdb_50.predict(hdb50_X_test,
        batch_size=batch_size)
1113.     hdb50_trainPredict2=sc.fit_transform(trainPredict_hdb50)
1114.     hdb50_trainPredict=sc.inverse_transform(hdb50_trainPredict2)
1115.     hdb50_trainY2 = sc.fit_transform([hdb50_y_train])
1116.     hdb50_trainY = sc.inverse_transform(hdb50_trainY2)
1117.     hdb50_testPredict2 = sc.fit_transform(testPredict_hdb50)
1118.     hdb50_testPredict = sc.inverse_transform(hdb50_testPredict2)
1119.     hdb50_testY2 = sc.fit_transform([hdb50_y_test])
1120.     hdb50_testY = sc.inverse_transform(hdb50_testY2)

1121.     cnn_history_hdb_50=cnn_model_hdb_50.fit(hdb50_X_train,hdb50_y_t
        rain,epochs=500,batch_size=64,validation_data=(hdb50_X_test,
        hdb50_y_test),callbacks=[earlyStop])
1122.     batch_size = 1
1123.     cnn_trainPredict_hdb50      =
        cnn_model_hdb_50.predict(hdb50_X_train, batch_size=batch_size)
1124.     cnn_model_hdb_50.reset_states()
1125.     cnn_testPredict_hdb50 = cnn_model_hdb_50.predict(hdb50_X_test,
        batch_size=batch_size)
1126.     cnn_hdb50_trainPredict2=sc.fit_transform(cnn_trainPredict_hdb50
        )
1127.     cnn_hdb50_trainPredict=sc.inverse_transform(cnn_hdb50_trainPred
        ict2)
1128.     cnn_hdb50_trainY2 = sc.fit_transform([hdb50_y_train])
1129.     cnn_hdb50_trainY = sc.inverse_transform(cnn_hdb50_trainY2)
1130.     cnn_hdb50_testPredict2      =
        sc.fit_transform(cnn_testPredict_hdb50)
1131.     cnn_hdb50_testPredict      =
        sc.inverse_transform(cnn_hdb50_testPredict2)
1132.     cnn_hdb50_testY2 = sc.fit_transform([hdb50_y_test])
1133.     cnn_hdb50_testY = sc.inverse_transform(cnn_hdb50_testY2)

1134.     lst = [
1135.         ['%.3f' % mae(hdb50_y_test, hdb50_linear_reg_pred),
        '%.3f' % mae(hdb50_y_test, hdb50_lasso_reg_pred), '%.3f' %
        mae(hdb50_y_test, hdb50_gaussian_pred), '%.3f' % mae(hdb50_y_test,
        hdb50_testPredict[:,0]), '%.3f' % mae(hdb50_y_test,
        cnn_hdb50_testPredict[:,0])],
1136.         ['%.3f' % rmse(hdb50_y_test, hdb50_linear_reg_pred),
        '%.3f' % rmse(hdb50_y_test, hdb50_lasso_reg_pred), '%.3f' %
        rmse(hdb50_y_test, hdb50_gaussian_pred), '%.3f' % rmse(hdb50_y_test,
        hdb50_testPredict[:,0]), '%.3f' % rmse(hdb50_y_test,
        cnn_hdb50_testPredict[:,0])],
1137.         ['%.3f' % mape(hdb50_y_test, hdb50_linear_reg_pred),
        '%.3f' % mape(hdb50_y_test, hdb50_lasso_reg_pred), '%.3f' %
        mape(hdb50_y_test, hdb50_gaussian_pred), '%.3f' % mape(hdb50_y_test,
        hdb50_testPredict[:,0]), '%.3f' % mape(hdb50_y_test,
        cnn_hdb50_testPredict[:,0])]
1138.     ]
1139.     df = pd.DataFrame(lst,
1140.         columns =['Linear Regression', 'Lasso
        Regression', 'LSSVM', 'LSTM', 'CNN'],
1141.         index=['MAE', 'RMSE', 'MAPE'])
1142.     df

```

```

1143.     history_ry_50=model_ry_50.fit(ry50_X_train,ry50_y_train,epochs=
        500,batch_size=64,validation_data=(ry50_X_test,
        ry50_y_test),callbacks=[earlyStop])
1144.     batch_size = 1
1145.     trainPredict_ry50      =      model_ry_50.predict(ry50_X_train,
        batch_size=batch_size)
1146.     model_ry_50.reset_states()
1147.     testPredict_ry50      =      model_ry_50.predict(ry50_X_test,
        batch_size=batch_size)
1148.     ry50_trainPredict2=sc.fit_transform(trainPredict_ry50)
1149.     ry50_trainPredict=sc.inverse_transform(ry50_trainPredict2)
1150.     ry50_trainY2 = sc.fit_transform([ry50_y_train])
1151.     ry50_trainY = sc.inverse_transform(ry50_trainY2)
1152.     ry50_testPredict2 = sc.fit_transform(testPredict_ry50)
1153.     ry50_testPredict = sc.inverse_transform(ry50_testPredict2)
1154.     ry50_testY2 = sc.fit_transform([ry50_y_test])
1155.     ry50_testY = sc.inverse_transform(ry50_testY2)

1156.     cnn_history_ry_50=cnn_model_ry_50.fit(ry50_X_train,ry50_y_train
        ,epochs=500,batch_size=64,validation_data=(ry50_X_test,
        ry50_y_test),callbacks=[earlyStop])
1157.     batch_size = 1
1158.     cnn_trainPredict_ry50 = cnn_model_ry_50.predict(ry50_X_train,
        batch_size=batch_size)
1159.     cnn_model_ry_50.reset_states()
1160.     cnn_testPredict_ry50 = cnn_model_ry_50.predict(ry50_X_test,
        batch_size=batch_size)
1161.     cnn_ry50_trainPredict2=sc.fit_transform(cnn_trainPredict_ry50)
1162.     cnn_ry50_trainPredict=sc.inverse_transform(cnn_ry50_trainPredic
        t2)
1163.     cnn_ry50_trainY2 = sc.fit_transform([ry50_y_train])
1164.     cnn_ry50_trainY = sc.inverse_transform(cnn_ry50_trainY2)
1165.     cnn_ry50_testPredict2 = sc.fit_transform(cnn_testPredict_ry50)
1166.     cnn_ry50_testPredict = sc.inverse_transform(cnn_ry50_testPredict2)
1167.     cnn_ry50_testY2 = sc.fit_transform([ry50_y_test])
1168.     cnn_ry50_testY = sc.inverse_transform(cnn_ry50_testY2)

1169.     lst = [
1170.         ['%.3f' % mae(ry50_y_test, ry50_linear_reg_pred), '%.3f'
        % mae(ry50_y_test, ry50_lasso_reg_pred), '%.3f' % mae(ry50_y_test,
        ry50_gaussian_pred), '%.3f' % mae(ry50_y_test, ry50_testPredict[:,0]),
        '%.3f' % mae(ry50_y_test, cnn_ry50_testPredict[:,0])],
1171.         ['%.3f' % rmse(ry50_y_test, ry50_linear_reg_pred),
        '%.3f' % rmse(ry50_y_test, ry50_lasso_reg_pred), '%.3f' %
        rmse(ry50_y_test, ry50_gaussian_pred), '%.3f' % rmse(ry50_y_test,
        ry50_testPredict[:,0]), '%.3f' % rmse(ry50_y_test,
        cnn_ry50_testPredict[:,0])],
1172.         ['%.3f' % mape(ry50_y_test, ry50_linear_reg_pred),
        '%.3f' % mape(ry50_y_test, ry50_lasso_reg_pred), '%.3f' %
        mape(ry50_y_test, ry50_gaussian_pred), '%.3f' % mape(ry50_y_test,
        ry50_testPredict[:,0]), '%.3f' % mape(ry50_y_test,
        cnn_ry50_testPredict[:,0])]
1173.     ]
1174.     df = pd.DataFrame(lst,
1175.         columns      =['Linear      Regression',      'Lasso
        Regression', 'LSSVM', 'LSTM', 'CNN'],
1176.         index=['MAE', 'RMSE', 'MAPE'])

```

1177. df

EVALUATION TESTING 60%

```
1178. history_bac_60=model_bac_60.fit(bac60_X_train,bac60_y_train,epoch
    chs=500,batch_size=64,validation_data=(bac60_X_test,
    bac60_y_test),callbacks=[earlyStop])
1179. batch_size = 1
1180. trainPredict_bac60 = model_bac_60.predict(bac60_X_train,
    batch_size=batch_size)
1181. model_bac_60.reset_states()
1182. testPredict_bac60 = model_bac_60.predict(bac60_X_test,
    batch_size=batch_size)
1183. bac60_trainPredict2=sc.fit_transform(trainPredict_bac60)
1184. bac60_trainPredict=sc.inverse_transform(bac60_trainPredict2)
1185. bac60_trainY2 = sc.fit_transform([bac60_y_train])
1186. bac60_trainY = sc.inverse_transform(bac60_trainY2)
1187. bac60_testPredict2 = sc.fit_transform(testPredict_bac60)
1188. bac60_testPredict = sc.inverse_transform(bac60_testPredict2)
1189. bac60_testY2 = sc.fit_transform([bac60_y_test])
1190. bac60_testY = sc.inverse_transform(bac60_testY2)

1191. cnn_history_bac_60=cnn_model_bac_60.fit(bac60_X_train,bac60_y_t
    rain,epochs=500,batch_size=64,validation_data=(bac60_X_test,
    bac60_y_test),callbacks=[earlyStop])
1192. batch_size = 1
1193. cnn_trainPredict_bac60 =
    cnn_model_bac_60.predict(bac60_X_train, batch_size=batch_size)
1194. cnn_model_bac_60.reset_states()
1195. cnn_testPredict_bac60 = cnn_model_bac_60.predict(bac60_X_test,
    batch_size=batch_size)
1196. cnn_bac60_trainPredict2=sc.fit_transform(cnn_trainPredict_bac60
    )
1197. cnn_bac60_trainPredict=sc.inverse_transform(cnn_bac60_trainPred
    ict2)
1198. cnn_bac60_trainY2 = sc.fit_transform([bac60_y_train])
1199. cnn_bac60_trainY = sc.inverse_transform(cnn_bac60_trainY2)
1200. cnn_bac60_testPredict2 =
    sc.fit_transform(cnn_testPredict_bac60)
1201. cnn_bac60_testPredict =
    sc.inverse_transform(cnn_bac60_testPredict2)
1202. cnn_bac60_testY2 = sc.fit_transform([bac60_y_test])
1203. cnn_bac60_testY = sc.inverse_transform(cnn_bac60_testY2)

1204. lst = [
1205.     ['%.3f' % mae(bac60_y_test, bac60_linear_reg_pred),
    '%.3f' % mae(bac60_y_test, bac60_lasso_reg_pred), '%.3f' %
    mae(bac60_y_test, bac60_gaussian_pred), '%.3f' % mae(bac60_y_test,
    bac60_testPredict[:,0]), '%.3f' % mae(bac60_y_test,
    cnn_bac60_testPredict[:,0])],
1206.     ['%.3f' % rmse(bac60_y_test, bac60_linear_reg_pred),
    '%.3f' % rmse(bac60_y_test, bac60_lasso_reg_pred), '%.3f' %
    rmse(bac60_y_test, bac60_gaussian_pred), '%.3f' % rmse(bac60_y_test,
    bac60_testPredict[:,0]), '%.3f' % rmse(bac60_y_test,
    cnn_bac60_testPredict[:,0])],
1207.     ['%.3f' % mape(bac60_y_test, bac60_linear_reg_pred),
    '%.3f' % mape(bac60_y_test, bac60_lasso_reg_pred), '%.3f' %
    mape(bac60_y_test, bac60_gaussian_pred), '%.3f' % mape(bac60_y_test,
    bac60_testPredict[:,0]), '%.3f' % mape(bac60_y_test,
    cnn_bac60_testPredict[:,0])]
```

```

1208.     ]
1209.     df = pd.DataFrame(lst,
1210.         columns = ['Linear Regression', 'Lasso
Regression', 'LSSVM', 'LSTM', 'CNN'],
1211.         index=['MAE', 'RMSE', 'MAPE'])
1212.     df

1213.     history_hdb_60=model_hdb_60.fit(hdb60_X_train,hdb60_y_train,epo
chs=500,batch_size=64,validation_data=(hdb60_X_test,
hdb60_y_test),callbacks=[earlyStop])
1214.     batch_size = 1
1215.     trainPredict_hdb60 = model_hdb_60.predict(hdb60_X_train,
batch_size=batch_size)
1216.     model_hdb_60.reset_states()
1217.     testPredict_hdb60 = model_hdb_60.predict(hdb60_X_test,
batch_size=batch_size)
1218.     hdb60_trainPredict2=sc.fit_transform(trainPredict_hdb60)
1219.     hdb60_trainPredict=sc.inverse_transform(hdb60_trainPredict2)
1220.     hdb60_trainY2 = sc.fit_transform([hdb60_y_train])
1221.     hdb60_trainY = sc.inverse_transform(hdb60_trainY2)
1222.     hdb60_testPredict2 = sc.fit_transform(testPredict_hdb60)
1223.     hdb60_testPredict = sc.inverse_transform(hdb60_testPredict2)
1224.     hdb60_testY2 = sc.fit_transform([hdb60_y_test])
1225.     hdb60_testY = sc.inverse_transform(hdb60_testY2)

1226.     cnn_history_hdb_60=cnn_model_hdb_60.fit(hdb60_X_train,hdb60_y_t
rain,epochs=500,batch_size=64,validation_data=(hdb60_X_test,
hdb60_y_test),callbacks=[earlyStop])
1227.     batch_size = 1
1228.     cnn_trainPredict_hdb60 =
cnn_model_hdb_60.predict(hdb60_X_train, batch_size=batch_size)
1229.     cnn_model_hdb_60.reset_states()
1230.     cnn_testPredict_hdb60 = cnn_model_hdb_60.predict(hdb60_X_test,
batch_size=batch_size)
1231.     cnn_hdb60_trainPredict2=sc.fit_transform(cnn_trainPredict_hdb60
)
1232.     cnn_hdb60_trainPredict=sc.inverse_transform(cnn_hdb60_trainPred
ict2)
1233.     cnn_hdb60_trainY2 = sc.fit_transform([hdb60_y_train])
1234.     cnn_hdb60_trainY = sc.inverse_transform(cnn_hdb60_trainY2)
1235.     cnn_hdb60_testPredict2 =
sc.fit_transform(cnn_testPredict_hdb60)
1236.     cnn_hdb60_testPredict =
sc.inverse_transform(cnn_hdb60_testPredict2)
1237.     cnn_hdb60_testY2 = sc.fit_transform([hdb60_y_test])
1238.     cnn_hdb60_testY = sc.inverse_transform(cnn_hdb60_testY2)

1239.     lst = [
1240.         ['%.3f' % mae(hdb60_y_test, hdb60_linear_reg_pred),
%.3f' % mae(hdb60_y_test, hdb60_lasso_reg_pred), '%.3f' %
mae(hdb60_y_test, hdb60_gaussian_pred), '%.3f' % mae(hdb60_y_test,
hdb60_testPredict[:,0]), '%.3f' % mae(hdb60_y_test,
cnn_hdb60_testPredict[:,0])],
1241.         ['%.3f' % rmse(hdb60_y_test, hdb60_linear_reg_pred),
%.3f' % rmse(hdb60_y_test, hdb60_lasso_reg_pred), '%.3f' %
rmse(hdb60_y_test, hdb60_gaussian_pred), '%.3f' % rmse(hdb60_y_test,
hdb60_testPredict[:,0]), '%.3f' % rmse(hdb60_y_test,
cnn_hdb60_testPredict[:,0])],

```



```

1242.         ['%.3f' % mape(hdb60_y_test, hdb60_linear_reg_pred),
'%.3f' % mape(hdb60_y_test, hdb60_lasso_reg_pred), '%.3f' %
mape(hdb60_y_test, hdb60_gaussian_pred), '%.3f' % mape(hdb60_y_test,
hdb60_testPredict[:,0]), '%.3f' % mape(hdb60_y_test,
cnn_hdb60_testPredict[:,0])]
1243.     ]
1244.     df = pd.DataFrame(lst,
1245.         columns =['Linear Regression', 'Lasso
Regression', 'LSSVM', 'LSTM', 'CNN'],
1246.         index=['MAE', 'RMSE', 'MAPE'])
1247.     df

1248.     history_ry_60=model_ry_60.fit(ry60_X_train,ry60_y_train,epochs=
500,batch_size=64,validation_data=(ry60_X_test,
ry60_y_test),callbacks=[earlyStop])
1249.     batch_size = 1
1250.     trainPredict_ry60 = model_ry_60.predict(ry60_X_train,
batch_size=batch_size)
1251.     model_ry_60.reset_states()
1252.     testPredict_ry60 = model_ry_60.predict(ry60_X_test,
batch_size=batch_size)
1253.     ry60_trainPredict2=sc.fit_transform(trainPredict_ry60)
1254.     ry60_trainPredict=sc.inverse_transform(ry60_trainPredict2)
1255.     ry60_trainY2 = sc.fit_transform([ry60_y_train])
1256.     ry60_trainY = sc.inverse_transform(ry60_trainY2)
1257.     ry60_testPredict2 = sc.fit_transform(testPredict_ry60)
1258.     ry60_testPredict = sc.inverse_transform(ry60_testPredict2)
1259.     ry60_testY2 = sc.fit_transform([ry60_y_test])
1260.     ry60_testY = sc.inverse_transform(ry60_testY2)

1261.     cnn_history_ry_60=cnn_model_ry_60.fit(ry60_X_train,ry60_y_train
,epochs=500,batch_size=64,validation_data=(ry60_X_test,
ry60_y_test),callbacks=[earlyStop])
1262.     batch_size = 1
1263.     cnn_trainPredict_ry60 = cnn_model_ry_60.predict(ry60_X_train,
batch_size=batch_size)
1264.     cnn_model_ry_60.reset_states()
1265.     cnn_testPredict_ry60 = cnn_model_ry_60.predict(ry60_X_test,
batch_size=batch_size)
1266.     cnn_ry60_trainPredict2=sc.fit_transform(cnn_trainPredict_ry60)
1267.     cnn_ry60_trainPredict=sc.inverse_transform(cnn_ry60_trainPredic
t2)
1268.     cnn_ry60_trainY2 = sc.fit_transform([ry60_y_train])
1269.     cnn_ry60_trainY = sc.inverse_transform(cnn_ry60_trainY2)
1270.     cnn_ry60_testPredict2 = sc.fit_transform(cnn_testPredict_ry60)
1271.     cnn_ry60_testPredict =
sc.inverse_transform(cnn_ry60_testPredict2)
1272.     cnn_ry60_testY2 = sc.fit_transform([ry60_y_test])
1273.     cnn_ry60_testY = sc.inverse_transform(cnn_ry60_testY2)

1274.     lst = [
1275.         ['%.3f' % mae(ry60_y_test, ry60_linear_reg_pred), '%.3f'
% mae(ry60_y_test, ry60_lasso_reg_pred), '%.3f' % mae(ry60_y_test,
ry60_gaussian_pred), '%.3f' % mae(ry60_y_test, ry60_testPredict[:,0]),
'%.3f' % mae(ry60_y_test, cnn_ry60_testPredict[:,0])],
1276.         ['%.3f' % rmse(ry60_y_test, ry60_linear_reg_pred),
'%.3f' % rmse(ry60_y_test, ry60_lasso_reg_pred), '%.3f' %
rmse(ry60_y_test, ry60_gaussian_pred), '%.3f' % rmse(ry60_y_test,

```

```

    ry60_testPredict[:,0]),          '%.3f'          %          rmse(ry60_y_test,
    cnn_ry60_testPredict[:,0])),
1277.          ['%.3f' % mape(ry60_y_test, ry60_linear_reg_pred),
    '%.3f' % mape(ry60_y_test, ry60_lasso_reg_pred), '%.3f' %
    mape(ry60_y_test, ry60_gaussian_pred), '%.3f' % mape(ry60_y_test,
    ry60_testPredict[:,0]),          '%.3f'          %          mape(ry60_y_test,
    cnn_ry60_testPredict[:,0])]
1278.          ]
1279.          df = pd.DataFrame(lst,
1280.          columns          =['Linear          Regression',          'Lasso
    Regression', 'LSSVM', 'LSTM', 'CNN'],
1281.          index=['MAE', 'RMSE', 'MAPE'])
1282.          df

```

EVALUATION TESTING 80%

```

1283.          history_bac_80=model_bac_80.fit(bac80_X_train,bac80_y_train,epo
    chs=500,batch_size=64,validation_data=(bac80_X_test,
    bac80_y_test),callbacks=[earlyStop])
1284.          batch_size = 1
1285.          trainPredict_bac80          =          model_bac_80.predict(bac80_X_train,
    batch_size=batch_size)
1286.          model_bac_80.reset_states()
1287.          testPredict_bac80          =          model_bac_80.predict(bac80_X_test,
    batch_size=batch_size)
1288.          bac80_trainPredict2=sc.fit_transform(trainPredict_bac80)
1289.          bac80_trainPredict=sc.inverse_transform(bac80_trainPredict2)
1290.          bac80_trainY2 = sc.fit_transform([bac80_y_train])
1291.          bac80_trainY = sc.inverse_transform(bac80_trainY2)
1292.          bac80_testPredict2 = sc.fit_transform(testPredict_bac80)
1293.          bac80_testPredict = sc.inverse_transform(bac80_testPredict2)
1294.          bac80_testY2 = sc.fit_transform([bac80_y_test])
1295.          bac80_testY = sc.inverse_transform(bac80_testY2)

1296.          cnn_history_bac_80=cnn_model_bac_80.fit(bac80_X_train,bac80_y_t
    rain,epochs=500,batch_size=64,validation_data=(bac80_X_test,
    bac80_y_test),callbacks=[earlyStop])
1297.          batch_size = 1
1298.          cnn_trainPredict_bac80          =
    cnn_model_bac_80.predict(bac80_X_train, batch_size=batch_size)
1299.          cnn_model_bac_80.reset_states()
1300.          cnn_testPredict_bac80 = cnn_model_bac_80.predict(bac80_X_test,
    batch_size=batch_size)
1301.          cnn_bac80_trainPredict2=sc.fit_transform(cnn_trainPredict_bac80
    )
1302.          cnn_bac80_trainPredict=sc.inverse_transform(cnn_bac80_trainPred
    ict2)
1303.          cnn_bac80_trainY2 = sc.fit_transform([bac80_y_train])
1304.          cnn_bac80_trainY = sc.inverse_transform(cnn_bac80_trainY2)
1305.          cnn_bac80_testPredict2          =
    sc.fit_transform(cnn_testPredict_bac80)
1306.          cnn_bac80_testPredict          =
    sc.inverse_transform(cnn_bac80_testPredict2)
1307.          cnn_bac80_testY2 = sc.fit_transform([bac80_y_test])
1308.          cnn_bac80_testY = sc.inverse_transform(cnn_bac80_testY2)

1309.          lst = [
1310.          ['%.3f' % mae(bac80_y_test, bac80_linear_reg_pred),
    '%.3f' % mae(bac80_y_test, bac80_lasso_reg_pred), '%.3f' %
    mae(bac80_y_test, bac80_gaussian_pred), '%.3f' % mae(bac80_y_test,

```

```

    bac80_testPredict[:,0]),          '%.3f'          %          mae(bac80_y_test,
    cnn_bac80_testPredict[:,0]))],
1311.          ['%.3f' % rmse(bac80_y_test, bac80_linear_reg_pred),
'%.3f' % rmse(bac80_y_test, bac80_lasso_reg_pred), '%.3f' %
rmse(bac80_y_test, bac80_gaussian_pred), '%.3f' % rmse(bac80_y_test,
bac80_testPredict[:,0]),          '%.3f'          %          rmse(bac80_y_test,
cnn_bac80_testPredict[:,0]))],
1312.          ['%.3f' % mape(bac80_y_test, bac80_linear_reg_pred),
'%.3f' % mape(bac80_y_test, bac80_lasso_reg_pred), '%.3f' %
mape(bac80_y_test, bac80_gaussian_pred), '%.3f' % mape(bac80_y_test,
bac80_testPredict[:,0]),          '%.3f'          %          mape(bac80_y_test,
cnn_bac80_testPredict[:,0))]
1313.          ]
1314.  df = pd.DataFrame(lst,
1315.                    columns =['Linear      Regression',      'Lasso
Regression', 'LSSVM', 'LSTM', 'CNN'],
1316.                    index=['MAE', 'RMSE', 'MAPE'])
1317.  df

1318.  history_hdb_80=model_hdb_80.fit(hdb80_X_train,hdb80_y_train, epo
chs=500,batch_size=64,validation_data=(hdb80_X_test,
hdb80_y_test),callbacks=[earlyStop])
1319.  batch_size = 1
1320.  trainPredict_hdb80      =      model_hdb_80.predict(hdb80_X_train,
batch_size=batch_size)
1321.  model_hdb_80.reset_states()
1322.  testPredict_hdb80      =      model_hdb_80.predict(hdb80_X_test,
batch_size=batch_size)
1323.  hdb80_trainPredict2=sc.fit_transform(trainPredict_hdb80)
1324.  hdb80_trainPredict=sc.inverse_transform(hdb80_trainPredict2)
1325.  hdb80_trainY2 = sc.fit_transform([hdb80_y_train])
1326.  hdb80_trainY = sc.inverse_transform(hdb80_trainY2)
1327.  hdb80_testPredict2 = sc.fit_transform(testPredict_hdb80)
1328.  hdb80_testPredict = sc.inverse_transform(hdb80_testPredict2)
1329.  hdb80_testY2 = sc.fit_transform([hdb80_y_test])
1330.  hdb80_testY = sc.inverse_transform(hdb80_testY2)

1331.  cnn_history_hdb_80=cnn_model_hdb_80.fit(hdb80_X_train,hdb80_y_t
rain,epochs=500,batch_size=64,validation_data=(hdb80_X_test,
hdb80_y_test),callbacks=[earlyStop])
1332.  batch_size = 1
1333.  cnn_trainPredict_hdb80      =
cnn_model_hdb_80.predict(hdb80_X_train, batch_size=batch_size)
1334.  cnn_model_hdb_80.reset_states()
1335.  cnn_testPredict_hdb80 = cnn_model_hdb_80.predict(hdb80_X_test,
batch_size=batch_size)
1336.  cnn_hdb80_trainPredict2=sc.fit_transform(cnn_trainPredict_hdb80
)
1337.  cnn_hdb80_trainPredict=sc.inverse_transform(cnn_hdb80_trainPred
ict2)
1338.  cnn_hdb80_trainY2 = sc.fit_transform([hdb80_y_train])
1339.  cnn_hdb80_trainY = sc.inverse_transform(cnn_hdb80_trainY2)
1340.  cnn_hdb80_testPredict2      =
sc.fit_transform(cnn_testPredict_hdb80)
1341.  cnn_hdb80_testPredict      =
sc.inverse_transform(cnn_hdb80_testPredict2)
1342.  cnn_hdb80_testY2 = sc.fit_transform([hdb80_y_test])
1343.  cnn_hdb80_testY = sc.inverse_transform(cnn_hdb80_testY2)

```

```

1344.     lst = [
1345.         ['%.3f' % mae(hdb80_y_test, hdb80_linear_reg_pred),
%.3f' % mae(hdb80_y_test, hdb80_lasso_reg_pred), '%.3f' %
mae(hdb80_y_test, hdb80_gaussian_pred), '%.3f' % mae(hdb80_y_test,
hdb80_testPredict[:,0]), '%.3f' % mae(hdb80_y_test,
cnn_hdb80_testPredict[:,0])],
1346.         ['%.3f' % rmse(hdb80_y_test, hdb80_linear_reg_pred),
%.3f' % rmse(hdb80_y_test, hdb80_lasso_reg_pred), '%.3f' %
rmse(hdb80_y_test, hdb80_gaussian_pred), '%.3f' % rmse(hdb80_y_test,
hdb80_testPredict[:,0]), '%.3f' % rmse(hdb80_y_test,
cnn_hdb80_testPredict[:,0])],
1347.         ['%.3f' % mape(hdb80_y_test, hdb80_linear_reg_pred),
%.3f' % mape(hdb80_y_test, hdb80_lasso_reg_pred), '%.3f' %
mape(hdb80_y_test, hdb80_gaussian_pred), '%.3f' % mape(hdb80_y_test,
hdb80_testPredict[:,0]), '%.3f' % mape(hdb80_y_test,
cnn_hdb80_testPredict[:,0])]
1348.     ]
1349.     df = pd.DataFrame(lst,
1350.         columns = ['Linear Regression', 'Lasso
Regression', 'LSSVM', 'LSTM', 'CNN'],
1351.         index=['MAE', 'RMSE', 'MAPE'])
1352.     df

1353.     history_ry_80=model_ry_80.fit(ry80_X_train,ry80_y_train,epochs=
500,batch_size=64,validation_data=(ry80_X_test,
ry80_y_test),callbacks=[earlyStop])
1354.     batch_size = 1
1355.     trainPredict_ry80 = model_ry_80.predict(ry80_X_train,
batch_size=batch_size)
1356.     model_ry_80.reset_states()
1357.     testPredict_ry80 = model_ry_80.predict(ry80_X_test,
batch_size=batch_size)
1358.     ry80_trainPredict2=sc.fit_transform(trainPredict_ry80)
1359.     ry80_trainPredict=sc.inverse_transform(ry80_trainPredict2)
1360.     ry80_trainY2 = sc.fit_transform([ry80_y_train])
1361.     ry80_trainY = sc.inverse_transform(ry80_trainY2)
1362.     ry80_testPredict2 = sc.fit_transform(testPredict_ry80)
1363.     ry80_testPredict = sc.inverse_transform(ry80_testPredict2)
1364.     ry80_testY2 = sc.fit_transform([ry80_y_test])
1365.     ry80_testY = sc.inverse_transform(ry80_testY2)

1366.     cnn_history_ry_80=cnn_model_ry_80.fit(ry80_X_train,ry80_y_train
,epochs=500,batch_size=64,validation_data=(ry80_X_test,
ry80_y_test),callbacks=[earlyStop])
1367.     batch_size = 1
1368.     cnn_trainPredict_ry80 = cnn_model_ry_80.predict(ry80_X_train,
batch_size=batch_size)
1369.     cnn_model_ry_80.reset_states()
1370.     cnn_testPredict_ry80 = cnn_model_ry_80.predict(ry80_X_test,
batch_size=batch_size)
1371.     cnn_ry80_trainPredict2=sc.fit_transform(cnn_trainPredict_ry80)
1372.     cnn_ry80_trainPredict=sc.inverse_transform(cnn_ry80_trainPredic
t2)
1373.     cnn_ry80_trainY2 = sc.fit_transform([ry80_y_train])
1374.     cnn_ry80_trainY = sc.inverse_transform(cnn_ry80_trainY2)
1375.     cnn_ry80_testPredict2 = sc.fit_transform(cnn_testPredict_ry80)

```

```

1376.     cnn_ry80_testPredict                                     =
         sc.inverse_transform(cnn_ry80_testPredict2)
1377.     cnn_ry80_testY2 = sc.fit_transform([ry80_y_test])
1378.     cnn_ry80_testY = sc.inverse_transform(cnn_ry80_testY2)

1379.     lst = [
1380.         ['%.3f' % mae(ry80_y_test, ry80_linear_reg_pred), '%.3f'
         % mae(ry80_y_test, ry80_lasso_reg_pred), '%.3f' % mae(ry80_y_test,
         ry80_gaussian_pred), '%.3f' % mae(ry80_y_test, ry80_testPredict[:,0]),
         '%.3f' % mae(ry80_y_test, cnn_ry80_testPredict[:,0])],
1381.         ['%.3f' % rmse(ry80_y_test, ry80_linear_reg_pred),
         '%.3f' % rmse(ry80_y_test, ry80_lasso_reg_pred), '%.3f' %
         rmse(ry80_y_test, ry80_gaussian_pred), '%.3f' % rmse(ry80_y_test,
         ry80_testPredict[:,0]), '%.3f' % rmse(ry80_y_test,
         cnn_ry80_testPredict[:,0])],
1382.         ['%.3f' % mape(ry80_y_test, ry80_linear_reg_pred),
         '%.3f' % mape(ry80_y_test, ry80_lasso_reg_pred), '%.3f' %
         mape(ry80_y_test, ry80_gaussian_pred), '%.3f' % mape(ry80_y_test,
         ry80_testPredict[:,0]), '%.3f' % mape(ry80_y_test,
         cnn_ry80_testPredict[:,0])]
1383.     ]
1384.     df = pd.DataFrame(lst,
1385.         columns = ['Linear Regression', 'Lasso
         Regression', 'LSSVM', 'LSTM', 'CNN'],
1386.         index=['MAE', 'RMSE', 'MAPE'])
1387.     df

```



PAPER NAME

TA-19.K1.0046.docx

WORD COUNT

7173 Words

CHARACTER COUNT

46570 Characters

PAGE COUNT

39 Pages

FILE SIZE

3.5MB

SUBMISSION DATE

Dec 15, 2022 1:29 PM GMT+7

REPORT DATE

Dec 15, 2022 1:30 PM GMT+7

● **14% Overall Similarity**

The combined total of all matches, including overlapping sources, for each database.

- 7% Internet database
- 8% Publications database
- Crossref database
- Crossref Posted Content database
- 13% Submitted Works database

● **Excluded from Similarity Report**

- Bibliographic material
- Quoted material
- Cited material
- Small Matches (Less than 10 words)

