APPENDIX

DOWNLOAD AND UNZIP DATASET

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

IMPORT LIBRARIES

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import math
import tensorflow as tf
import warnings

from keras.callbacks import EarlyStopping
from sklearn import metrics
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.base import BaseEstimator, RegressorMixin
from itertools import combinations_with_replacement
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import LSTM, Activation, Dense, Dropout, RepeatVector, Flatten
from keras.layers.convolutional import Conv1D, MaxPooling1D

%matplotlib inline
warnings.filterwarnings('ignore')
```

IMPORT LIBRARIES

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

FUNCTION HEATMAP

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

VISUALIZE DATA BAC

```
bac.head(10)

bac = bac.astype('float64')
```

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

bac.info()

bac.describe()

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

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

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

show_heatmap(bac)

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

bac['Open:30 days rolling']=bac['Open'].rolling(30).mean()
bac['High:30 days rolling']=bac['High'].rolling(30).mean()
bac['Low:30 days rolling']=bac['Low'].rolling(30).mean()
bac['Close:30 days rolling']=bac['Close'].rolling(30).mean()
bac['Adj Close:30 days rolling']=bac['Adj Close'].rolling(30).mean()
bac['Volume:30 days rolling']=bac['Volume'].rolling(30).mean()

bac[['Open','Open:30 days rolling']].plot(figsize=(12,5))
bac[['High','High:30 days rolling']].plot(figsize=(12,5))
bac[['Low','Low:30 days rolling']].plot(figsize=(12,5))
bac[['Close','Close:30 days rolling']].plot(figsize=(12,5))
bac[['Adj Close','Adj Close:30 days rolling']].plot(figsize=(12,5))
bac[['Volume','Volume:30 days rolling']].plot(figsize=(12,5))

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

bac['delta_open_close_day_before_%']=((bac['Open']-
bac['Close'].shift(1))/bac['Close'].shift(1))*100
order = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday"]
box_plot    =    sns.boxplot(x=pd.to_datetime(bac.index).day_name(),
y=bac['delta_open_close_day_before_%'], order=order)

ax = box_plot.axes
lines = ax.get_lines()
categories = ax.get_xticks()

for cat in categories:
    y = round(lines[4+cat*6].get_ydata()[0],1)

    ax.text(
        cat,
        y,
        f'{y}',
        ha='center',
        va='center',
        fontweight='bold',
```

```
            size=10,
            color='white',
            bbox=dict(facecolor='#445A64'))

box_plot.figure.tight_layout()
```

VISUALIZE DATA HDB

```
hdb.head(10)

hdb = hdb.astype('float64')

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

hdb.info()

hdb.describe()

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

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

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

show_heatmap(hdb)

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

hdb['Open:30 days rolling']=hdb['Open'].rolling(30).mean()
hdb['High:30 days rolling']=hdb['High'].rolling(30).mean()
hdb['Low:30 days rolling']=hdb['Low'].rolling(30).mean()
hdb['Close:30 days rolling']=hdb['Close'].rolling(30).mean()
hdb['Adj Close:30 days rolling']=hdb['Adj Close'].rolling(30).mean()
hdb['Volume:30 days rolling']=hdb['Volume'].rolling(30).mean()

hdb[['Open','Open:30 days rolling']].plot(figsize=(12,5))
hdb[['High','High:30 days rolling']].plot(figsize=(12,5))
hdb[['Low','Low:30 days rolling']].plot(figsize=(12,5))
hdb[['Close','Close:30 days rolling']].plot(figsize=(12,5))
hdb[['Adj Close','Adj Close:30 days rolling']].plot(figsize=(12,5))
hdb[['Volume','Volume:30 days rolling']].plot(figsize=(12,5))

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

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

ax = box_plot.axes
```

```
lines = ax.get_lines()
categories = ax.get_xticks()

for cat in categories:
    y = round(lines[4+cat*6].get_ydata()[0],1)

    ax.text(
        cat,
        y,
        f'{y}',
        ha='center',
        va='center',
        fontweight='bold',
        size=10,
        color='white',
        bbox=dict(facecolor='#445A64'))

box_plot.figure.tight_layout()
```

VISUALIZE DATA RY

```
ry.head(10)

ry = ry.astype('float64')

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

ry.info()

ry.describe()

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

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

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

show_heatmap(ry)

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

ry['Open:30 days rolling']=ry['Open'].rolling(30).mean()
ry['High:30 days rolling']=ry['High'].rolling(30).mean()
ry['Low:30 days rolling']=ry['Low'].rolling(30).mean()
ry['Close:30 days rolling']=ry['Close'].rolling(30).mean()
ry['Adj Close:30 days rolling']=ry['Adj Close'].rolling(30).mean()
ry['Volume:30 days rolling']=ry['Volume'].rolling(30).mean()

ry[['Open','Open:30 days rolling']].plot(figsize=(12,5))
ry[['High','High:30 days rolling']].plot(figsize=(12,5))
ry[['Low','Low:30 days rolling']].plot(figsize=(12,5))
```

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

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

ry['delta_open_close_day_before_%']=((ry['Open']-
   ry['Close'].shift(1))/ry['Close'].shift(1))*100
order = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday"]
box_plot     =     sns.boxplot(x=pd.to_datetime(ry.index).day_name(),
   y=ry['delta_open_close_day_before_%'], order=order)

ax = box_plot.axes
lines = ax.get_lines()
categories = ax.get_xticks()

for cat in categories:
    y = round(lines[4+cat*6].get_ydata()[0],1)

    ax.text(
        cat,
        y,
        f'{y}',
        ha='center',
        va='center',
        fontweight='bold',
        size=10,
        color='white',
        bbox=dict(facecolor='#445A64'))

box_plot.figure.tight_layout()
```

FUNCTION EVALUATION

```
def rmse(y_test, y_pred):
  rmse = np.sqrt(mean_squared_error(y_test, y_pred))
  return rmse

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

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

FEATURE SCALING

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

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

ry = ry[['HL_PCT', 'PCT_change', 'Adj Close','Volume']]
```

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

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

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

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

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

PREPARING DATA

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

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

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

SPLIT TRAIN DATA AND TEST DATA

```
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)
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)
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)

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)
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)
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)

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)
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)
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)

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)
```

```
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)
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)

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)
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)
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

```
class Regression(object):
    def __init__(self, n_iterations, learning_rate):
        self.m = None
        self.n = None
        self.w = None
        self.b = None
        self.n_iterations = n_iterations
        self.learning_rate = learning_rate

    def cost_function(self, y, y_pred):
        return (1 / (2*self.m)) * np.sum(np.square(y_pred - y)) + self.regularization(self.w)

    def hypothesis(self, weights, X):
        return np.dot(X, weights)

    def initialize_weights(self, n_features):
        limit = 1 / math.sqrt(n_features)
        self.w = np.random.uniform(-limit, limit, (n_features, ))

    def fit(self, X, y):
        X = np.insert(X, 0, 1, axis=1)
        self.training_errors = []
        self.initialize_weights(n_features=X.shape[1])

        self.m = X.shape[0]
        self.n = X.shape[1]

        self.b = 0

        for i in range(self.n_iterations):
            y_pred = self.hypothesis(self.w, X)
            cost = self.cost_function(y, y_pred)
            self.training_errors.append(cost)
            grad_w = -(y - y_pred).dot(X) + self.regularization.grad(self.w)
            self.w = self.w - self.learning_rate * grad_w

    def predict(self, X):
        X = np.insert(X, 0, 1, axis=1)
        y_pred = self.hypothesis(self.w, X)
        return y_pred
```

BUILD LINEAR REGRESSION

```
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,
340.                                               learning_rate)
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.
```

```
    kernels = {'rbf': rbf}
    return kernels[name]

  def __OptimizeParams(self):
    Omega = self.kernel_(self.x, self.x)
    Ones = np.array([[1]]*len(self.y))

    A_dag = np.linalg.pinv(np.block([
      [0,                              Ones.T                          ],
      [Ones,   Omega + self.gamma**-1 * np.identity(len(self.y))]
    ]))
    B = np.concatenate((np.array([0]), self.y), axis=None)

    solution = np.dot(A_dag, B)
    self.intercept_ = solution[0]
    self.coef_      = solution[1:]

  def fit(self, X: np.ndarray, y: np.ndarray):
    if isinstance(X, (pd.DataFrame, pd.Series)):
      Xloc = X.to_numpy()
    else:
      Xloc = X

    if isinstance(y, (pd.DataFrame, pd.Series)):
      yloc = y.to_numpy()
    else:
      yloc = y

    if (Xloc.ndim == 2) and (yloc.ndim == 1):
      self.x = Xloc
      self.y = yloc
      self.__OptimizeParams()
    else:
      message = "The fit procedure requires a 2D numpy array of
features "\
        "and 1D array of targets"
      raise Exception(message)

  def predict(self, X: np.ndarray)->np.ndarray:
    Ker = self.kernel_(X, self.x)
    Y = np.dot(self.coef_, Ker.T) + self.intercept_
    return Y
```

BUILD LSTM FOR DATASET BAC

```
model_bac_20= Sequential()
model_bac_20.add(LSTM(units=100,return_sequences=True,
  activation='relu', input_shape=(bac20_X_train.shape[1], 1)))
model_bac_20.add(Dropout(rate=0.2))
model_bac_20.add(LSTM(units=100,return_sequences=True,
  activation='relu'))
model_bac_20.add(Dropout(rate=0.01))
model_bac_20.add(LSTM(units=100,return_sequences=True,
  activation='relu'))
model_bac_20.add(Dropout(rate=0.002))
model_bac_20.add(LSTM(units=100,return_sequences=True,
  activation='relu'))
model_bac_20.add(Dropout(rate=0.02))
model_bac_20.add(LSTM(units=100, activation='relu'))
model_bac_20.add(Dropout(rate=0.02))
```

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

model_bac_20.summary()

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

model_bac_40= Sequential()
model_bac_40.add(LSTM(units=100,return_sequences=True,
   activation='relu', input_shape=(bac40_X_train.shape[1], 1)))
model_bac_40.add(Dropout(rate=0.2))
model_bac_40.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_bac_40.add(Dropout(rate=0.01))
model_bac_40.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_bac_40.add(Dropout(rate=0.002))
model_bac_40.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_bac_40.add(Dropout(rate=0.02))
model_bac_40.add(LSTM(units=100, activation='relu'))
model_bac_40.add(Dropout(rate=0.02))
model_bac_40.add(Dense(units=100))
model_bac_40.compile(loss='mse',  optimizer='adam',  metrics=['mse',
   'mae', 'mape'])

model_bac_50= Sequential()
model_bac_50.add(LSTM(units=100,return_sequences=True,
   activation='relu', input_shape=(bac50_X_train.shape[1], 1)))
model_bac_50.add(Dropout(rate=0.2))
model_bac_50.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_bac_50.add(Dropout(rate=0.01))
model_bac_50.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_bac_50.add(Dropout(rate=0.002))
model_bac_50.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_bac_50.add(Dropout(rate=0.02))
model_bac_50.add(LSTM(units=100, activation='relu'))
model_bac_50.add(Dropout(rate=0.02))
model_bac_50.add(Dense(units=100))
model_bac_50.compile(loss='mse',  optimizer='adam',  metrics=['mse',
   'mae', 'mape'])

model_bac_60= Sequential()
model_bac_60.add(LSTM(units=100,return_sequences=True,
   activation='relu', input_shape=(bac60_X_train.shape[1], 1)))
model_bac_60.add(Dropout(rate=0.2))
model_bac_60.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_bac_60.add(Dropout(rate=0.01))
model_bac_60.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
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))
```

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

model_hdb_50= Sequential()
model_hdb_50.add(LSTM(units=100,return_sequences=True,
   activation='relu', input_shape=(hdb50_X_train.shape[1], 1)))
model_hdb_50.add(Dropout(rate=0.2))
model_hdb_50.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_hdb_50.add(Dropout(rate=0.01))
model_hdb_50.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_hdb_50.add(Dropout(rate=0.002))
model_hdb_50.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_hdb_50.add(Dropout(rate=0.02))
model_hdb_50.add(LSTM(units=100, activation='relu'))
model_hdb_50.add(Dropout(rate=0.02))
model_hdb_50.add(Dense(units=100))
model_hdb_50.compile(loss='mse',  optimizer='adam',  metrics=['mse',
   'mae', 'mape'])

model_hdb_60= Sequential()
model_hdb_60.add(LSTM(units=100,return_sequences=True,
   activation='relu', input_shape=(hdb60_X_train.shape[1], 1)))
model_hdb_60.add(Dropout(rate=0.2))
model_hdb_60.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_hdb_60.add(Dropout(rate=0.01))
model_hdb_60.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_hdb_60.add(Dropout(rate=0.002))
model_hdb_60.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_hdb_60.add(Dropout(rate=0.02))
model_hdb_60.add(LSTM(units=100, activation='relu'))
model_hdb_60.add(Dropout(rate=0.02))
model_hdb_60.add(Dense(units=100))
model_hdb_60.compile(loss='mse',  optimizer='adam',  metrics=['mse',
   'mae', 'mape'])

model_hdb_80= Sequential()
model_hdb_80.add(LSTM(units=100,return_sequences=True,
   activation='relu', input_shape=(hdb80_X_train.shape[1], 1)))
model_hdb_80.add(Dropout(rate=0.2))
model_hdb_80.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_hdb_80.add(Dropout(rate=0.01))
model_hdb_80.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_hdb_80.add(Dropout(rate=0.002))
model_hdb_80.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
```

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

BUILD LSTM FOR DATASET RY

```
model_ry_20= Sequential()
model_ry_20.add(LSTM(units=100,return_sequences=True,
   activation='relu', input_shape=(ry20_X_train.shape[1], 1)))
model_ry_20.add(Dropout(rate=0.2))
model_ry_20.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_ry_20.add(Dropout(rate=0.01))
model_ry_20.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_ry_20.add(Dropout(rate=0.002))
model_ry_20.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_ry_20.add(Dropout(rate=0.02))
model_ry_20.add(LSTM(units=100, activation='relu'))
model_ry_20.add(Dropout(rate=0.02))
model_ry_20.add(Dense(units=100))
model_ry_20.compile(loss='mse',  optimizer='adam',  metrics=['mse',
   'mae', 'mape'])

model_ry_40= Sequential()
model_ry_40.add(LSTM(units=100,return_sequences=True,
   activation='relu', input_shape=(ry40_X_train.shape[1], 1)))
model_ry_40.add(Dropout(rate=0.2))
model_ry_40.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_ry_40.add(Dropout(rate=0.01))
model_ry_40.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_ry_40.add(Dropout(rate=0.002))
model_ry_40.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_ry_40.add(Dropout(rate=0.02))
model_ry_40.add(LSTM(units=100, activation='relu'))
model_ry_40.add(Dropout(rate=0.02))
model_ry_40.add(Dense(units=100))
model_ry_40.compile(loss='mse',  optimizer='adam',  metrics=['mse',
   'mae', 'mape'])

model_ry_50= Sequential()
model_ry_50.add(LSTM(units=100,return_sequences=True,
   activation='relu', input_shape=(ry50_X_train.shape[1], 1)))
model_ry_50.add(Dropout(rate=0.2))
model_ry_50.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_ry_50.add(Dropout(rate=0.01))
model_ry_50.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_ry_50.add(Dropout(rate=0.002))
model_ry_50.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_ry_50.add(Dropout(rate=0.02))
```

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

model_ry_60= Sequential()
model_ry_60.add(LSTM(units=100,return_sequences=True,
   activation='relu', input_shape=(ry60_X_train.shape[1], 1)))
model_ry_60.add(Dropout(rate=0.2))
model_ry_60.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_ry_60.add(Dropout(rate=0.01))
model_ry_60.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_ry_60.add(Dropout(rate=0.002))
model_ry_60.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_ry_60.add(Dropout(rate=0.02))
model_ry_60.add(LSTM(units=100, activation='relu'))
model_ry_60.add(Dropout(rate=0.02))
model_ry_60.add(Dense(units=100))
model_ry_60.compile(loss='mse',  optimizer='adam',  metrics=['mse',
   'mae', 'mape'])

model_ry_80= Sequential()
model_ry_80.add(LSTM(units=100,return_sequences=True,
   activation='relu', input_shape=(ry80_X_train.shape[1], 1)))
model_ry_80.add(Dropout(rate=0.2))
model_ry_80.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_ry_80.add(Dropout(rate=0.01))
model_ry_80.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_ry_80.add(Dropout(rate=0.002))
model_ry_80.add(LSTM(units=100,return_sequences=True,
   activation='relu'))
model_ry_80.add(Dropout(rate=0.02))
model_ry_80.add(LSTM(units=100, activation='relu'))
model_ry_80.add(Dropout(rate=0.02))
model_ry_80.add(Dense(units=100))
model_ry_80.compile(loss='mse',  optimizer='adam',  metrics=['mse',
   'mae', 'mape'])
```

BUILD CNN FOR DATASET BAC

```
cnn_model_bac_20 = Sequential()
cnn_model_bac_20.add(Conv1D(filters=64,                kernel_size=2,
   activation='relu', input_shape=(bac20_X_train.shape[1], 1)))
cnn_model_bac_20.add(MaxPooling1D(pool_size=2))
cnn_model_bac_20.add(Conv1D(filters=32,                kernel_size=1,
   activation='relu'))
cnn_model_bac_20.add(MaxPooling1D(pool_size=1))
cnn_model_bac_20.add(Conv1D(filters=64,                kernel_size=1,
   activation='relu'))
cnn_model_bac_20.add(MaxPooling1D(pool_size=1))
cnn_model_bac_20.add(Conv1D(filters=32,                kernel_size=1,
   activation='relu'))
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'])
```

```
cnn_model_bac_60 = Sequential()
cnn_model_bac_60.add(Conv1D(filters=64,                kernel_size=2,
   activation='relu', input_shape=(bac60_X_train.shape[1], 1)))
cnn_model_bac_60.add(MaxPooling1D(pool_size=2))
cnn_model_bac_60.add(Conv1D(filters=32,                kernel_size=1,
   activation='relu'))
cnn_model_bac_60.add(MaxPooling1D(pool_size=1))
cnn_model_bac_60.add(Conv1D(filters=64,                kernel_size=1,
   activation='relu'))
cnn_model_bac_60.add(MaxPooling1D(pool_size=1))
cnn_model_bac_60.add(Conv1D(filters=32,                kernel_size=1,
   activation='relu'))
cnn_model_bac_60.add(MaxPooling1D(pool_size=1))
cnn_model_bac_60.add(Conv1D(filters=64,                kernel_size=1,
   activation='relu'))
cnn_model_bac_60.add(MaxPooling1D(pool_size=1))
cnn_model_bac_60.add(Flatten())
cnn_model_bac_60.add(Dense(50, activation='relu'))
cnn_model_bac_60.add(Dense(1))
cnn_model_bac_60.compile(loss='mse',              optimizer='adam',
   metrics=['mse', 'mae', 'mape'])

cnn_model_bac_80 = Sequential()
cnn_model_bac_80.add(Conv1D(filters=64,                kernel_size=2,
   activation='relu', input_shape=(bac80_X_train.shape[1], 1)))
cnn_model_bac_80.add(MaxPooling1D(pool_size=2))
cnn_model_bac_80.add(Conv1D(filters=32,                kernel_size=1,
   activation='relu'))
cnn_model_bac_80.add(MaxPooling1D(pool_size=1))
cnn_model_bac_80.add(Conv1D(filters=64,                kernel_size=1,
   activation='relu'))
cnn_model_bac_80.add(MaxPooling1D(pool_size=1))
cnn_model_bac_80.add(Conv1D(filters=32,                kernel_size=1,
   activation='relu'))
cnn_model_bac_80.add(MaxPooling1D(pool_size=1))
cnn_model_bac_80.add(Conv1D(filters=64,                kernel_size=1,
   activation='relu'))
cnn_model_bac_80.add(MaxPooling1D(pool_size=1))
cnn_model_bac_80.add(Flatten())
cnn_model_bac_80.add(Dense(50, activation='relu'))
cnn_model_bac_80.add(Dense(1))
cnn_model_bac_80.compile(loss='mse',              optimizer='adam',
   metrics=['mse', 'mae', 'mape'])
```

BUILD CNN FOR DATASET HDB

```
cnn_model_hdb_20 = Sequential()
cnn_model_hdb_20.add(Conv1D(filters=64,                kernel_size=2,
   activation='relu', input_shape=(hdb20_X_train.shape[1], 1)))
cnn_model_hdb_20.add(MaxPooling1D(pool_size=2))
cnn_model_hdb_20.add(Conv1D(filters=32,                kernel_size=1,
   activation='relu'))
cnn_model_hdb_20.add(MaxPooling1D(pool_size=1))
cnn_model_hdb_20.add(Conv1D(filters=64,                kernel_size=1,
   activation='relu'))
cnn_model_hdb_20.add(MaxPooling1D(pool_size=1))
cnn_model_hdb_20.add(Conv1D(filters=32,                kernel_size=1,
   activation='relu'))
cnn_model_hdb_20.add(MaxPooling1D(pool_size=1))
```

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

cnn_model_hdb_40 = Sequential()
cnn_model_hdb_40.add(Conv1D(filters=64,                kernel_size=2,
   activation='relu', input_shape=(hdb40_X_train.shape[1], 1)))
cnn_model_hdb_40.add(MaxPooling1D(pool_size=2))
cnn_model_hdb_40.add(Conv1D(filters=32,                kernel_size=1,
   activation='relu'))
cnn_model_hdb_40.add(MaxPooling1D(pool_size=1))
cnn_model_hdb_40.add(Conv1D(filters=64,                kernel_size=1,
   activation='relu'))
cnn_model_hdb_40.add(MaxPooling1D(pool_size=1))
cnn_model_hdb_40.add(Conv1D(filters=32,                kernel_size=1,
   activation='relu'))
cnn_model_hdb_40.add(MaxPooling1D(pool_size=1))
cnn_model_hdb_40.add(Conv1D(filters=64,                kernel_size=1,
   activation='relu'))
cnn_model_hdb_40.add(MaxPooling1D(pool_size=1))
cnn_model_hdb_40.add(Flatten())
cnn_model_hdb_40.add(Dense(50, activation='relu'))
cnn_model_hdb_40.add(Dense(1))
cnn_model_hdb_40.compile(loss='mse',               optimizer='adam',
   metrics=['mse', 'mae', 'mape'])

cnn_model_hdb_50 = Sequential()
cnn_model_hdb_50.add(Conv1D(filters=64,                kernel_size=2,
   activation='relu', input_shape=(hdb50_X_train.shape[1], 1)))
cnn_model_hdb_50.add(MaxPooling1D(pool_size=2))
cnn_model_hdb_50.add(Conv1D(filters=32,                kernel_size=1,
   activation='relu'))
cnn_model_hdb_50.add(MaxPooling1D(pool_size=1))
cnn_model_hdb_50.add(Conv1D(filters=64,                kernel_size=1,
   activation='relu'))
cnn_model_hdb_50.add(MaxPooling1D(pool_size=1))
cnn_model_hdb_50.add(Conv1D(filters=32,                kernel_size=1,
   activation='relu'))
cnn_model_hdb_50.add(MaxPooling1D(pool_size=1))
cnn_model_hdb_50.add(Conv1D(filters=64,                kernel_size=1,
   activation='relu'))
cnn_model_hdb_50.add(MaxPooling1D(pool_size=1))
cnn_model_hdb_50.add(Flatten())
cnn_model_hdb_50.add(Dense(50, activation='relu'))
cnn_model_hdb_50.add(Dense(1))
cnn_model_hdb_50.compile(loss='mse',               optimizer='adam',
   metrics=['mse', 'mae', 'mape'])

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

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

cnn_model_ry_40 = Sequential()
cnn_model_ry_40.add(Conv1D(filters=64,                kernel_size=2,
   activation='relu', input_shape=(ry40_X_train.shape[1], 1)))
cnn_model_ry_40.add(MaxPooling1D(pool_size=2))
cnn_model_ry_40.add(Conv1D(filters=32,                kernel_size=1,
   activation='relu'))
cnn_model_ry_40.add(MaxPooling1D(pool_size=1))
cnn_model_ry_40.add(Conv1D(filters=64,                kernel_size=1,
   activation='relu'))
cnn_model_ry_40.add(MaxPooling1D(pool_size=1))
cnn_model_ry_40.add(Conv1D(filters=32,                kernel_size=1,
   activation='relu'))
cnn_model_ry_40.add(MaxPooling1D(pool_size=1))
cnn_model_ry_40.add(Conv1D(filters=64,                kernel_size=1,
   activation='relu'))
cnn_model_ry_40.add(MaxPooling1D(pool_size=1))
cnn_model_ry_40.add(Flatten())
cnn_model_ry_40.add(Dense(50, activation='relu'))
cnn_model_ry_40.add(Dense(1))
cnn_model_ry_40.compile(loss='mse',                optimizer='adam',
   metrics=['mse', 'mae', 'mape'])

cnn_model_ry_50 = Sequential()
cnn_model_ry_50.add(Conv1D(filters=64,                kernel_size=2,
   activation='relu', input_shape=(ry50_X_train.shape[1], 1)))
cnn_model_ry_50.add(MaxPooling1D(pool_size=2))
cnn_model_ry_50.add(Conv1D(filters=32,                kernel_size=1,
   activation='relu'))
cnn_model_ry_50.add(MaxPooling1D(pool_size=1))
cnn_model_ry_50.add(Conv1D(filters=64,                kernel_size=1,
   activation='relu'))
cnn_model_ry_50.add(MaxPooling1D(pool_size=1))
cnn_model_ry_50.add(Conv1D(filters=32,                kernel_size=1,
   activation='relu'))
cnn_model_ry_50.add(MaxPooling1D(pool_size=1))
cnn_model_ry_50.add(Conv1D(filters=64,                kernel_size=1,
   activation='relu'))
cnn_model_ry_50.add(MaxPooling1D(pool_size=1))
cnn_model_ry_50.add(Flatten())
cnn_model_ry_50.add(Dense(50, activation='relu'))
cnn_model_ry_50.add(Dense(1))
cnn_model_ry_50.compile(loss='mse',                optimizer='adam',
   metrics=['mse', 'mae', 'mape'])

cnn_model_ry_60 = Sequential()
cnn_model_ry_60.add(Conv1D(filters=64,                kernel_size=2,
   activation='relu', input_shape=(ry60_X_train.shape[1], 1)))
cnn_model_ry_60.add(MaxPooling1D(pool_size=2))
cnn_model_ry_60.add(Conv1D(filters=32,                kernel_size=1,
   activation='relu'))
cnn_model_ry_60.add(MaxPooling1D(pool_size=1))
cnn_model_ry_60.add(Conv1D(filters=64,                kernel_size=1,
   activation='relu'))
cnn_model_ry_60.add(MaxPooling1D(pool_size=1))
cnn_model_ry_60.add(Conv1D(filters=32,                kernel_size=1,
   activation='relu'))
```

```
cnn_model_ry_60.add(MaxPooling1D(pool_size=1))
cnn_model_ry_60.add(Conv1D(filters=64, kernel_size=1,
activation='relu'))
cnn_model_ry_60.add(MaxPooling1D(pool_size=1))
cnn_model_ry_60.add(Flatten())
cnn_model_ry_60.add(Dense(50, activation='relu'))
cnn_model_ry_60.add(Dense(1))
cnn_model_ry_60.compile(loss='mse', optimizer='adam',
metrics=['mse', 'mae', 'mape'])

cnn_model_ry_80 = Sequential()
cnn_model_ry_80.add(Conv1D(filters=64, kernel_size=2,
activation='relu', input_shape=(ry80_X_train.shape[1], 1)))
cnn_model_ry_80.add(MaxPooling1D(pool_size=2))
cnn_model_ry_80.add(Conv1D(filters=32, kernel_size=1,
activation='relu'))
cnn_model_ry_80.add(MaxPooling1D(pool_size=1))
cnn_model_ry_80.add(Conv1D(filters=64, kernel_size=1,
activation='relu'))
cnn_model_ry_80.add(MaxPooling1D(pool_size=1))
cnn_model_ry_80.add(Conv1D(filters=32, kernel_size=1,
activation='relu'))
cnn_model_ry_80.add(MaxPooling1D(pool_size=1))
cnn_model_ry_80.add(Conv1D(filters=64, kernel_size=1,
activation='relu'))
cnn_model_ry_80.add(MaxPooling1D(pool_size=1))
cnn_model_ry_80.add(Flatten())
cnn_model_ry_80.add(Dense(50, activation='relu'))
cnn_model_ry_80.add(Dense(1))
cnn_model_ry_80.compile(loss='mse', optimizer='adam',
metrics=['mse', 'mae', 'mape'])
```

EARLY STOPPING

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

EVALUATION TESTING 20%

```
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])
batch_size = 1
trainPredict_bac20 = model_bac_20.predict(bac20_X_train,
batch_size=batch_size)
model_bac_20.reset_states()
testPredict_bac20 = model_bac_20.predict(bac20_X_test,
batch_size=batch_size)
bac20_trainPredict2=sc.fit_transform(trainPredict_bac20)
bac20_trainPredict=sc.inverse_transform(bac20_trainPredict2)
bac20_trainY2 = sc.fit_transform([bac20_y_train])
bac20_trainY = sc.inverse_transform(bac20_trainY2)
bac20_testPredict2 = sc.fit_transform(testPredict_bac20)
bac20_testPredict = sc.inverse_transform(bac20_testPredict2)
bac20_testY2 = sc.fit_transform([bac20_y_test])
bac20_testY = sc.inverse_transform(bac20_testY2)

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])
batch_size = 1
```

```
cnn_trainPredict_bac20 = cnn_model_bac_20.predict(bac20_X_train, batch_size=batch_size)
cnn_model_bac_20.reset_states()
cnn_testPredict_bac20 = cnn_model_bac_20.predict(bac20_X_test, batch_size=batch_size)
cnn_bac20_trainPredict2=sc.fit_transform(cnn_trainPredict_bac20)
cnn_bac20_trainPredict=sc.inverse_transform(cnn_bac20_trainPredict2)
cnn_bac20_trainY2 = sc.fit_transform([bac20_y_train])
cnn_bac20_trainY = sc.inverse_transform(cnn_bac20_trainY2)
cnn_bac20_testPredict2 = sc.fit_transform(cnn_testPredict_bac20)
cnn_bac20_testPredict = sc.inverse_transform(cnn_bac20_testPredict2)
cnn_bac20_testY2 = sc.fit_transform([bac20_y_test])
cnn_bac20_testY = sc.inverse_transform(cnn_bac20_testY2)

lst = [
        ['%.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])],
        ['%.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])],
        ['%.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])]
    ]
df = pd.DataFrame(lst,
                  columns =['Linear Regression', 'Lasso Regression', 'LSSVM', 'LSTM', 'CNN'],
                  index=['MAE', 'RMSE', 'MAPE'])
df


history_hdb_20=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])
batch_size = 1
trainPredict_hdb20 = model_hdb_20.predict(hdb20_X_train, batch_size=batch_size)
model_hdb_20.reset_states()
testPredict_hdb20 = model_hdb_20.predict(hdb20_X_test, batch_size=batch_size)
hdb20_trainPredict2=sc.fit_transform(trainPredict_hdb20)
hdb20_trainPredict=sc.inverse_transform(hdb20_trainPredict2)
hdb20_trainY2 = sc.fit_transform([hdb20_y_train])
hdb20_trainY = sc.inverse_transform(hdb20_trainY2)
hdb20_testPredict2 = sc.fit_transform(testPredict_hdb20)
hdb20_testPredict = sc.inverse_transform(hdb20_testPredict2)
hdb20_testY2 = sc.fit_transform([hdb20_y_test])
hdb20_testY = sc.inverse_transform(hdb20_testY2)

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,epoc
   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)
```

```
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])
batch_size = 1
cnn_trainPredict_bac40 = cnn_model_bac_40.predict(bac40_X_train,batch_size=batch_size)
cnn_model_bac_40.reset_states()
cnn_testPredict_bac40 = cnn_model_bac_40.predict(bac40_X_test,batch_size=batch_size)
cnn_bac40_trainPredict2=sc.fit_transform(cnn_trainPredict_bac40)
cnn_bac40_trainPredict=sc.inverse_transform(cnn_bac40_trainPredict2)
cnn_bac40_trainY2 = sc.fit_transform([bac40_y_train])
cnn_bac40_trainY = sc.inverse_transform(cnn_bac40_trainY2)
cnn_bac40_testPredict2 = sc.fit_transform(cnn_testPredict_bac40)
cnn_bac40_testPredict = sc.inverse_transform(cnn_bac40_testPredict2)
cnn_bac40_testY2 = sc.fit_transform([bac40_y_test])
cnn_bac40_testY = sc.inverse_transform(cnn_bac40_testY2)

lst = [
        ['%.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])],
        ['%.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])],
        ['%.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])]
    ]
df = pd.DataFrame(lst,
                  columns=['Linear Regression', 'Lasso Regression', 'LSSVM', 'LSTM', 'CNN'],
                  index=['MAE', 'RMSE', 'MAPE'])
df


history_hdb_40=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])
batch_size = 1
trainPredict_bac40 = model_hdb_40.predict(hdb40_X_train,batch_size=batch_size)
model_hdb_40.reset_states()
testPredict_bac40 = model_hdb_40.predict(hdb40_X_test,batch_size=batch_size)
bac40_trainPredict2=sc.fit_transform(trainPredict_bac40)
hdb40_trainPredict=sc.inverse_transform(bac40_trainPredict2)
bac40_trainY2 = sc.fit_transform([hdb40_y_train])
hdb40_trainY = sc.inverse_transform(bac40_trainY2)
bac40_testPredict2 = sc.fit_transform(testPredict_bac40)
hdb40_testPredict = sc.inverse_transform(bac40_testPredict2)
bac40_testY2 = sc.fit_transform([hdb40_y_test])
```

```
hdb40_testY = sc.inverse_transform(bac40_testY2)

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])
batch_size = 1
cnn_trainPredict_hdb40 = cnn_model_hdb_40.predict(hdb40_X_train, batch_size=batch_size)
cnn_model_hdb_40.reset_states()
cnn_testPredict_hdb40 = cnn_model_hdb_40.predict(hdb40_X_test, batch_size=batch_size)
cnn_hdb40_trainPredict2=sc.fit_transform(cnn_trainPredict_hdb40)
cnn_hdb40_trainPredict=sc.inverse_transform(cnn_hdb40_trainPredict2)
cnn_hdb40_trainY2 = sc.fit_transform([hdb40_y_train])
cnn_hdb40_trainY = sc.inverse_transform(cnn_hdb40_trainY2)
cnn_hdb40_testPredict2 = sc.fit_transform(cnn_testPredict_hdb40)
cnn_hdb40_testPredict = sc.inverse_transform(cnn_hdb40_testPredict2)
cnn_hdb40_testY2 = sc.fit_transform([hdb40_y_test])
cnn_hdb40_testY = sc.inverse_transform(cnn_hdb40_testY2)

lst = [
            ['%.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])],
            ['%.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])],
            ['%.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])]
        ]
df = pd.DataFrame(lst,
                  columns=['Linear Regression', 'Lasso Regression', 'LSSVM', 'LSTM', 'CNN'],
                  index=['MAE', 'RMSE', 'MAPE'])
df


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])
batch_size = 1
trainPredict_ry40 = model_ry_40.predict(ry40_X_train, batch_size=batch_size)
model_ry_40.reset_states()
testPredict_ry40 = model_ry_40.predict(ry40_X_test, batch_size=batch_size)
ry40_trainPredict2=sc.fit_transform(trainPredict_ry40)
ry40_trainPredict=sc.inverse_transform(ry40_trainPredict2)
```

```
ry40_trainY2 = sc.fit_transform([ry40_y_train])
ry40_trainY = sc.inverse_transform(ry40_trainY2)
ry40_testPredict2 = sc.fit_transform(testPredict_ry40)
ry40_testPredict = sc.inverse_transform(ry40_testPredict2)
ry40_testY2 = sc.fit_transform([ry40_y_test])
ry40_testY = sc.inverse_transform(ry40_testY2)

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])
batch_size = 1
cnn_trainPredict_ry40 = cnn_model_ry_40.predict(ry40_X_train,
batch_size=batch_size)
cnn_model_ry_40.reset_states()
cnn_testPredict_ry40 = cnn_model_ry_40.predict(ry40_X_test,
batch_size=batch_size)
cnn_ry40_trainPredict2=sc.fit_transform(cnn_trainPredict_ry40)
cnn_ry40_trainPredict=sc.inverse_transform(cnn_ry40_trainPredic
t2)
cnn_ry40_trainY2 = sc.fit_transform([ry40_y_train])
cnn_ry40_trainY = sc.inverse_transform(cnn_ry40_trainY2)
cnn_ry40_testPredict2 = sc.fit_transform(cnn_testPredict_ry40)
cnn_ry40_testPredict =
sc.inverse_transform(cnn_ry40_testPredict2)
cnn_ry40_testY2 = sc.fit_transform([ry40_y_test])
cnn_ry40_testY = sc.inverse_transform(cnn_ry40_testY2)

lst = [
            ['%.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])],
            ['%.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])],
            ['%.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])]
        ]
df = pd.DataFrame(lst,
                  columns=['Linear Regression', 'Lasso
Regression', 'LSSVM', 'LSTM', 'CNN'],
                  index=['MAE', 'RMSE', 'MAPE'])
df
```

EVALUATION TESTING 50%

```
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])
batch_size = 1
trainPredict_bac50 = model_bac_50.predict(bac50_X_train,
batch_size=batch_size)
model_bac_50.reset_states()
testPredict_bac50 = model_bac_50.predict(bac50_X_test,
batch_size=batch_size)
```

```
    bac50_trainPredict2=sc.fit_transform(trainPredict_bac50)
    bac50_trainPredict=sc.inverse_transform(bac50_trainPredict2)
    bac50_trainY2 = sc.fit_transform([bac50_y_train])
    bac50_trainY = sc.inverse_transform(bac50_trainY2)
    bac50_testPredict2 = sc.fit_transform(testPredict_bac50)
    bac50_testPredict = sc.inverse_transform(bac50_testPredict2)
    bac50_testY2 = sc.fit_transform([bac50_y_test])
    bac50_testY = sc.inverse_transform(bac50_testY2)

    cnn_history_bac_50=cnn_model_bac_50.fit(bac50_X_train,bac50_y_t
rain,epochs=500,batch_size=64,validation_data=(bac50_X_test,
bac50_y_test),callbacks=[earlyStop])
    batch_size = 1
    cnn_trainPredict_bac50                                          =
cnn_model_bac_50.predict(bac50_X_train, batch_size=batch_size)
    cnn_model_bac_50.reset_states()
    cnn_testPredict_bac50  =  cnn_model_bac_50.predict(bac50_X_test,
batch_size=batch_size)
    cnn_bac50_trainPredict2=sc.fit_transform(cnn_trainPredict_bac50
)
    cnn_bac50_trainPredict=sc.inverse_transform(cnn_bac50_trainPred
ict2)
    cnn_bac50_trainY2 = sc.fit_transform([bac50_y_train])
    cnn_bac50_trainY = sc.inverse_transform(cnn_bac50_trainY2)
    cnn_bac50_testPredict2                                          =
sc.fit_transform(cnn_testPredict_bac50)
    cnn_bac50_testPredict                                           =
sc.inverse_transform(cnn_bac50_testPredict2)
    cnn_bac50_testY2 = sc.fit_transform([bac50_y_test])
    cnn_bac50_testY = sc.inverse_transform(cnn_bac50_testY2)

    lst = [
            ['%.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])],
            ['%.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])],
            ['%.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])]
          ]
    df = pd.DataFrame(lst,
                      columns     =['Linear     Regression',     'Lasso
Regression', 'LSSVM', 'LSTM', 'CNN'],
                      index=['MAE', 'RMSE', 'MAPE'])
    df


    history_hdb_50=model_hdb_50.fit(hdb50_X_train,hdb50_y_train,epo
chs=500,batch_size=64,validation_data=(hdb50_X_test,
hdb50_y_test),callbacks=[earlyStop])
    batch_size = 1
```

```
trainPredict_hdb50 = model_hdb_50.predict(hdb50_X_train, batch_size=batch_size)
model_hdb_50.reset_states()
testPredict_hdb50 = model_hdb_50.predict(hdb50_X_test, batch_size=batch_size)
hdb50_trainPredict2=sc.fit_transform(trainPredict_hdb50)
hdb50_trainPredict=sc.inverse_transform(hdb50_trainPredict2)
hdb50_trainY2 = sc.fit_transform([hdb50_y_train])
hdb50_trainY = sc.inverse_transform(hdb50_trainY2)
hdb50_testPredict2 = sc.fit_transform(testPredict_hdb50)
hdb50_testPredict = sc.inverse_transform(hdb50_testPredict2)
hdb50_testY2 = sc.fit_transform([hdb50_y_test])
hdb50_testY = sc.inverse_transform(hdb50_testY2)

cnn_history_hdb_50=cnn_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])
batch_size = 1
cnn_trainPredict_hdb50 = cnn_model_hdb_50.predict(hdb50_X_train, batch_size=batch_size)
cnn_model_hdb_50.reset_states()
cnn_testPredict_hdb50 = cnn_model_hdb_50.predict(hdb50_X_test, batch_size=batch_size)
cnn_hdb50_trainPredict2=sc.fit_transform(cnn_trainPredict_hdb50)
cnn_hdb50_trainPredict=sc.inverse_transform(cnn_hdb50_trainPredict2)
cnn_hdb50_trainY2 = sc.fit_transform([hdb50_y_train])
cnn_hdb50_trainY = sc.inverse_transform(cnn_hdb50_trainY2)
cnn_hdb50_testPredict2 = sc.fit_transform(cnn_testPredict_hdb50)
cnn_hdb50_testPredict = sc.inverse_transform(cnn_hdb50_testPredict2)
cnn_hdb50_testY2 = sc.fit_transform([hdb50_y_test])
cnn_hdb50_testY = sc.inverse_transform(cnn_hdb50_testY2)

lst = [
        ['%.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])],
        ['%.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])],
        ['%.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])]
    ]
df = pd.DataFrame(lst,
                  columns=['Linear Regression', 'Lasso Regression', 'LSSVM', 'LSTM', 'CNN'],
                  index=['MAE', 'RMSE', 'MAPE'])
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'])
```

```
df
```

EVALUATION TESTING 60%

```
history_bac_60=model_bac_60.fit(bac60_X_train,bac60_y_train,epochs=500,batch_size=64,validation_data=(bac60_X_test, bac60_y_test),callbacks=[earlyStop])
batch_size = 1
trainPredict_bac60 = model_bac_60.predict(bac60_X_train, batch_size=batch_size)
model_bac_60.reset_states()
testPredict_bac60 = model_bac_60.predict(bac60_X_test, batch_size=batch_size)
bac60_trainPredict2=sc.fit_transform(trainPredict_bac60)
bac60_trainPredict=sc.inverse_transform(bac60_trainPredict2)
bac60_trainY2 = sc.fit_transform([bac60_y_train])
bac60_trainY = sc.inverse_transform(bac60_trainY2)
bac60_testPredict2 = sc.fit_transform(testPredict_bac60)
bac60_testPredict = sc.inverse_transform(bac60_testPredict2)
bac60_testY2 = sc.fit_transform([bac60_y_test])
bac60_testY = sc.inverse_transform(bac60_testY2)

cnn_history_bac_60=cnn_model_bac_60.fit(bac60_X_train,bac60_y_train,epochs=500,batch_size=64,validation_data=(bac60_X_test, bac60_y_test),callbacks=[earlyStop])
batch_size = 1
cnn_trainPredict_bac60 = cnn_model_bac_60.predict(bac60_X_train, batch_size=batch_size)
cnn_model_bac_60.reset_states()
cnn_testPredict_bac60 = cnn_model_bac_60.predict(bac60_X_test, batch_size=batch_size)
cnn_bac60_trainPredict2=sc.fit_transform(cnn_trainPredict_bac60)
cnn_bac60_trainPredict=sc.inverse_transform(cnn_bac60_trainPredict2)
cnn_bac60_trainY2 = sc.fit_transform([bac60_y_train])
cnn_bac60_trainY = sc.inverse_transform(cnn_bac60_trainY2)
cnn_bac60_testPredict2 = sc.fit_transform(cnn_testPredict_bac60)
cnn_bac60_testPredict = sc.inverse_transform(cnn_bac60_testPredict2)
cnn_bac60_testY2 = sc.fit_transform([bac60_y_test])
cnn_bac60_testY = sc.inverse_transform(cnn_bac60_testY2)

lst = [
            ['%.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])],
            ['%.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])],
            ['%.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)
```

```
    lst = [
            ['%.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])],
            ['%.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])],
            ['%.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])]
          ]
    df = pd.DataFrame(lst,
                      columns=['Linear Regression', 'Lasso Regression', 'LSSVM', 'LSTM', 'CNN'],
                      index=['MAE', 'RMSE', 'MAPE'])
    df


    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])
    batch_size = 1
    trainPredict_ry80 = model_ry_80.predict(ry80_X_train, batch_size=batch_size)
    model_ry_80.reset_states()
    testPredict_ry80 = model_ry_80.predict(ry80_X_test, batch_size=batch_size)
    ry80_trainPredict2=sc.fit_transform(trainPredict_ry80)
    ry80_trainPredict=sc.inverse_transform(ry80_trainPredict2)
    ry80_trainY2 = sc.fit_transform([ry80_y_train])
    ry80_trainY = sc.inverse_transform(ry80_trainY2)
    ry80_testPredict2 = sc.fit_transform(testPredict_ry80)
    ry80_testPredict = sc.inverse_transform(ry80_testPredict2)
    ry80_testY2 = sc.fit_transform([ry80_y_test])
    ry80_testY = sc.inverse_transform(ry80_testY2)

    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])
    batch_size = 1
    cnn_trainPredict_ry80 = cnn_model_ry_80.predict(ry80_X_train, batch_size=batch_size)
    cnn_model_ry_80.reset_states()
    cnn_testPredict_ry80 = cnn_model_ry_80.predict(ry80_X_test, batch_size=batch_size)
    cnn_ry80_trainPredict2=sc.fit_transform(cnn_trainPredict_ry80)
    cnn_ry80_trainPredict=sc.inverse_transform(cnn_ry80_trainPredict2)
    cnn_ry80_trainY2 = sc.fit_transform([ry80_y_train])
    cnn_ry80_trainY = sc.inverse_transform(cnn_ry80_trainY2)
    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	CHARACTER COUNT
7173 Words	**46570 Characters**
PAGE COUNT	FILE SIZE
39 Pages	**3.5MB**
SUBMISSION DATE	REPORT DATE
Dec 15, 2022 1:29 PM GMT+7	**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 then 10 words)