APPENDIX

```
1.     DROP TABLE "TBLDATE" cascade constraints;
2.     DROP TABLE "TBLINPUT" cascade constraints;
3.     DROP TABLE "TBLOUTPUT" cascade constraints;
4.     DROP TABLE "TBLTABLESPC" cascade constraints;
5.     DROP SEQUENCE "TBDATE_SEQ";
6.     DROP SEQUENCE "TBLINPUT_SEQ";
7.     DROP SEQUENCE "TBLOUTPUT_SEQ";
8.     DROP SEQUENCE "TBLTABLESPACE_SEQ";
9.     --------------------------------------------------------
10.    --  DDL for Schema
11.    --------------------------------------------------------
12.
13.      CREATE TABLE "TBLDATE"
14.       (      "ID_DATE" NUMBER,
15.              "TANGGAL" DATE
16.       )   NO INMEMORY ;
17.
18.      CREATE TABLE "TBLINPUT"
19.       (      "ID_INPUT" NUMBER,
20.              "ID_DATE" NUMBER,
21.              "ID_TBLSPC" NUMBER,
22.              "WAKTU" VARCHAR2(20 BYTE),
23.              "MAX_SIZE" NUMBER,
24.              "FREE_SPC" NUMBER,
25.              "ID_OUTPUT" NUMBER,
26.              "START_AT" NUMBER,
27.              "ROOT_START" NUMBER
28.       )   NO INMEMORY ;
29.
30.      CREATE TABLE "TBLOUTPUT"
31.       (      "ID_OUTPUT" NUMBER,
32.              "ID_DATE" NUMBER
33.       )   NO INMEMORY ;
34.
35.      CREATE TABLE "TBLTABLESPC"
36.       (      "ID_TBLSPC" NUMBER,
37.              "NAMA_TBLSPC" VARCHAR2(20 BYTE)
38.       )   NO INMEMORY ;
39.
40.    --------------------------------------------------------
41.    --  DDL for Sequence
42.    --------------------------------------------------------
43.
44.            CREATE   SEQUENCE     "TBDATE_SEQ"     MINVALUE   1   MAXVALUE
9999999999999999999999999999 INCREMENT BY 1 START WITH 1 CACHE 20 NOORDER
NOCYCLE  NOKEEP  NOSCALE  GLOBAL ;
45.
46.           CREATE   SEQUENCE     "TBLINPUT_SEQ"     MINVALUE   1   MAXVALUE
9999999999999999999999999999 INCREMENT BY 1 START WITH 1 CACHE 20 NOORDER
NOCYCLE  NOKEEP  NOSCALE  GLOBAL ;
47.
```

```
48.           CREATE   SEQUENCE     "TBLOUTPUT_SEQ"     MINVALUE   1   MAXVALUE
9999999999999999999999999999 INCREMENT BY 1 START WITH 1 CACHE 20 NOORDER
NOCYCLE  NOKEEP  NOSCALE  GLOBAL ;
49.
50.          CREATE   SEQUENCE    "TBLTABLESPACE_SEQ"    MINVALUE  1  MAXVALUE
9999999999999999999999999999 INCREMENT BY 1 START WITH 1 CACHE 20 NOORDER
NOCYCLE  NOKEEP  NOSCALE  GLOBAL ;
51.
52.      CREATE UNIQUE INDEX "TBLOUTPUT_PK" ON "TBLOUTPUT" ("ID_OUTPUT")
53.      ;
54.    --------------------------------------------------------
55.    --  DDL for Index
56.    --------------------------------------------------------
57.
58.      CREATE UNIQUE INDEX "TBDATE_PK" ON "TBLDATE" ("ID_DATE")
59.      ;
60.
61.      CREATE UNIQUE INDEX "TBLINPUT_PK" ON "TBLINPUT" ("ID_INPUT")
62.      ;
63.
64.      CREATE UNIQUE INDEX "TBLTABLESPACE_PK" ON "TBLTABLESPC" ("ID_TBLSPC")
65.      ;
66.    --------------------------------------------------------
67.    --  DDL for Trigger
68.    --------------------------------------------------------
69.
70.
71.      CREATE OR REPLACE EDITIONABLE TRIGGER "TBDATE_TRG"
72.    BEFORE INSERT ON "TBLDATE"
73.    FOR EACH ROW
74.    BEGIN
75.      <<COLUMN_SEQUENCES>>
76.      BEGIN
77.        IF INSERTING AND :NEW.ID_DATE IS NULL THEN
78.          SELECT TBDATE_SEQ.NEXTVAL INTO :NEW.ID_DATE FROM SYS.DUAL;
79.        END IF;
80.      END COLUMN_SEQUENCES;
81.    END;
82.    /
83.    ALTER TRIGGER "TBDATE_TRG" ENABLE;
84.
85.      CREATE OR REPLACE EDITIONABLE TRIGGER "TBLINPUT_TRG"
86.    BEFORE INSERT ON TBLINPUT
87.    FOR EACH ROW
88.    BEGIN
89.      <<COLUMN_SEQUENCES>>
90.      BEGIN
91.        IF INSERTING AND :NEW.ID_INPUT IS NULL THEN
92.          SELECT TBLINPUT_SEQ.NEXTVAL INTO :NEW.ID_INPUT FROM SYS.DUAL;
93.        END IF;
94.      END COLUMN_SEQUENCES;
95.    END;
96.    /
97.    ALTER TRIGGER "TBLINPUT_TRG" ENABLE;
98.
99.      CREATE OR REPLACE EDITIONABLE TRIGGER "TBLOUTPUT_TRG"
```

```
100.   BEFORE INSERT ON TBLOUTPUT
101.   FOR EACH ROW
102.   BEGIN
103.     <<COLUMN_SEQUENCES>>
104.     BEGIN
105.       IF INSERTING AND :NEW.ID_OUTPUT IS NULL THEN
106.             SELECT  TBLOUTPUT_SEQ.NEXTVAL  INTO  :NEW.ID_OUTPUT  FROM
SYS.DUAL;
107.       END IF;
108.     END COLUMN_SEQUENCES;
109.   END;
110.   /
111.   ALTER TRIGGER "TBLOUTPUT_TRG" ENABLE;
112.
113.     CREATE OR REPLACE EDITIONABLE TRIGGER "TBLTABLESPACE_TRG"
114.   BEFORE INSERT ON "TBLTABLESPC"
115.   FOR EACH ROW
116.   BEGIN
117.     <<COLUMN_SEQUENCES>>
118.     BEGIN
119.       IF INSERTING AND :NEW.ID_TBLSPC IS NULL THEN
120.           SELECT  TBLTABLESPACE_SEQ.NEXTVAL  INTO  :NEW.ID_TBLSPC  FROM
SYS.DUAL;
121.       END IF;
122.     END COLUMN_SEQUENCES;
123.   END;
124.   /
125.   ALTER TRIGGER "TBLTABLESPACE_TRG" ENABLE;
126.
127.   --------------------------------------------------------
128.   --  Constraints for Tables
129.   --------------------------------------------------------
130.
131.     ALTER TABLE "TBLDATE" MODIFY ("ID_DATE" NOT NULL ENABLE);
132.      ALTER  TABLE  "TBLDATE"  ADD  CONSTRAINT  "TBDATE_PK"  PRIMARY  KEY
("ID_DATE")
133.     USING INDEX  ENABLE;
134.
135.     ALTER TABLE "TBLINPUT" MODIFY ("ID_INPUT" NOT NULL ENABLE);
136.      ALTER TABLE "TBLINPUT" ADD CONSTRAINT "TBLINPUT_PK" PRIMARY KEY
("ID_INPUT")
137.     USING INDEX  ENABLE;
138.
139.     ALTER TABLE "TBLOUTPUT" MODIFY ("ID_OUTPUT" NOT NULL ENABLE);
140.      ALTER  TABLE  "TBLOUTPUT"  ADD  CONSTRAINT  "TBLOUTPUT_PK"  PRIMARY
KEY ("ID_OUTPUT")
141.     USING INDEX  ENABLE;
142.
143.     ALTER TABLE "TBLTABLESPC" MODIFY ("ID_TBLSPC" NOT NULL ENABLE);
144.       ALTER  TABLE  "TBLTABLESPC"  ADD  CONSTRAINT  "TBLTABLESPACE_PK"
PRIMARY KEY ("ID_TBLSPC")
145.     USING INDEX  ENABLE;
```

Code 8 : Dataset schema

I. Simulation Database

```
1.     root_start = datetime(2019,9,25)
2.     start_at = datetime(2019,9,25)
3.     tglwaktu = datetime(2019,9,25)
4.     tblspace = "APPTBLSPC"
5.     tblname = "TBLSIMULATION"
```

Code 9 : Set global variable

```
1.     for hari in range(0,40):
2.         for jam in range(9,22,3):
3.             dt = tglwaktu + timedelta(days=hari, hours=jam)
4.             gndata = GenData(root_start, start_at, dt, tblspace,
5.                              tblname)
6.             gtdata = GetData(root_start, start_at, dt, tblspace)
7.             print(gndata.dt.strftime("%A %d %B %Y"))
8.             total = gndata.totIterations()
9.             gtdata.init()
10.            print(f"Iterating {total} times")
```

Code 10 Loop over days and hour, declare class variable, get total rows to input, save basic dataset to schema that doesn't include anything about size

```
1.     def totIterations(self):
2.         baseIter = random.randrange(2000,3000)
3.         baseAdd = 0
4.         baseExtra = 0
5.         if self.dt.day in range(1,12) or self.dt.day in range(23,32):
6.             if self.dt.day in range(23,32):
7.                 if self.dt.month == 12 :
8.                     beda = datetime(self.dt.year + 1, 1, 13) - self.dt
9.                 else:
10.                    beda = datetime(self.dt.year, self.dt.month +1 , 13)
11.                            - self.dt
12.                beda = float(beda.days / 10)
13.            else:
14.                beda = float(( 13 - self.dt.day ) / 10)
15.            print(f'Beda : {beda} hari')
16.            baseAdd = int(beda * random.randrange(2500,3000))
17.            print("Tanggal muda")
18.            if self.dt.weekday() in range(5,7):
19.                print('weekend')
20.                """ Jika weekend """
21.                rng_list= ["A"]*60 + ["B"]*20 + ["C"]*20
22.                chosen = random.choice(rng_list)
23.                if chosen is "A":
24.                    baseExtra = random.randrange(6000,7001)
25.                elif chosen is "B":
26.                    baseExtra = random.randrange(5000,6001)
27.                else:
28.                    baseExtra = random.randrange(4000,5001)
```

```
            else:
                print('weekday')
                """ Jika weekday """
                rng_list= ["A"]*50 + ["B"]*25 + ["C"]*25
                chosen = random.choice(rng_list)
                if chosen is "A":
                    baseExtra = random.randrange(3500,4001)
                elif chosen is "B":
                    baseExtra = random.randrange(3000,3501)
                else:
                    baseExtra = random.randrange(2000,3001)
        else:
            print("Tanggal tua")
            selisih = abs(23-self.dt.day)
            baseAdd = selisih * random.randrange(200,401)
            if self.dt.weekday() in range(5,7):
                print('weekend')
                """ Jika weekend """
                rng_list= ["A"]*60 + ["B"]*20 + ["C"]*20
                chosen = random.choice(rng_list)
                if chosen is "A":
                    baseExtra = random.randrange(4000,4501)
                elif chosen is "B":
                    baseExtra = random.randrange(2000,3001)
                else:
                    baseExtra = random.randrange(100,1001)

        return baseIter+baseAdd+baseExtra
```

Code 11: Algorithm to count how much rows inserted to table simulation that belongs to target tablespace.

```
1.  def insertRow(self,word, word2=None):
2.      try:
3.          if word2 is not None:
4.              rng1 = random.randrange(1,1001)
5.              rng2 = random.randrange(1001, 2001)
6.              rng3 = random.randrange(2001, 10000)
7.              sql = """
8.                  INSERT INTO {0}(WORD, WORD2, RNG1, RNG2, RNG3)
9.                  VALUES (:word, :word2, :rng1, :rng2, :rng3)
10.             """
11.             in_param = {
12.                 'word' : word,
13.                 'word2' : word2,
14.                 'rng1' : rng1,
15.                 'rng2' : rng2,
16.                 'rng3' : rng3
17.             }
18.             c = self.orcl.execute(sql.format(self.tblname),
19.                                   in_param, True)
20.         else:
21.             sql = """
22.                 INSERT INTO TBLSIMULATION2(WORD)
23.                 VALUES (:word)
24.             """
25.             in_param = {
26.                 'word' : word
27.             }
28.             c = self.orcl.execute(sql,in_param,True)
29.         return True
30.     except cx_Oracle.DatabaseError as e:
31.         errorObj, = e.args
32.         print("Error Code:", errorObj.code)
33.         print("Error Message:", errorObj.message)
34.         if errorObj.code == 1653 or errorObj.code == 1654:
35.             print("Tablespace full")
36.         return False
```

Code 12: Function to insert data to table simulation, return False if tablespace full.

```
1.  def init(self):
2.      data_to_save = {}
3.      data_to_save['nama_tblspc'] = self.tablespace
4.      data_to_save['tanggalwaktu'] = self.d
5.      data_to_save['start_at'] = self.start_at
6.      data_to_save['root_start'] = self.root_start
7.      self.sds.init(self.orcl, data_to_save)
```

Code 13: Function to insert data to dataset schema without size (called in code 10 line 9)

```
1.  for i in range(total):
2.      if i%1000 is 0:
3.          print(i)
4.      word = gndata.chooseWords()
5.      word2 = gndata.chooseWords()
6.      flag = gndata.insertRow(word,word2)
7.  #   if flag and i%100 is 0:
8.  #       flag = gndata.insertRow(word)
9.      if flag is False:
10.         break
```

Code 14: Function to insert rows with random data to table simulation

```
1.  if flag is True:
2.      gndata.upSim()
3.      gtdata.upSize()
4.      gtdata.orcl.disconnect()
5.      gndata.disconnect()
6.  else:
7.      gtdata.start(flag)
8.      gndata.upSim()
9.      gtdata.orcl.disconnect()
10.     gndata.disconnect()
11.     break
```

Code 15: If tablespace not full, runline 2 to 5, if full run 7 to 11. Gndata.upSim() function updates all rows that just get inserted with null id_input with current id_input. GtData.upSize() and gtdata.start(flag) update the current input size (max_size and free_spc) after inserting data to simulation. The difference is that gtdata.start(flag) updates all past data that are empty with the date when the tablespace is full.

II. The system

(1) Mining dataset process

i. Settings process

```
kr = keyring()
cwd = os.path.dirname(os.path.realpath(__file__))+"/"
filename = cwd+"settings_storage/config.json"
if os.path.isfile(filename):
    print(f"{bcolors.OKBLUE}Configuration file found [{filename}]"
    f"{bcolors.ENDC}")
    with open(filename) as json_data_file:
        data = json.load(json_data_file)
    prmpt_change = query_yes_no("Change Configuration?")
    if prmpt_change:
        chng_conf(data)
        print(bcolors.OKGREEN+"Config file saved."+bcolors.ENDC)
    else:
        print(bcolors.OKGREEN+"Closing the program."+bcolors.ENDC)
else:
    print(f"{bcolors.FAIL}Configuration file not found "
        f"[{filename}]{bcolors.ENDC}")
    print(f"{bcolors.HEADER}Creating new config file {bcolors.ENDC}")
    print(f"{bcolors.HEADER}Usage manual : Square brackets ([]) and "
        f"upper case Y/N means default value if you don't provide any "
        f"input{bcolors.ENDC}")
    data = {}

    data = set_storage(data)

    data = set_db(data)

    data = set_thresh(data)

    data = set_alert(data)

    data["tracking"]=True

    print(f'{bcolors.HEADER}Tablespace tracking turned on,'
            f' launch this settings again or edit the {filename}'
            f' "tracking" variable to False to turn off tracking.\n'
            f'{bcolors.ENDC}')
    print(bcolors.OKBLUE+"Writing the config file"+bcolors.ENDC)

    with open(filename,'w') as outfile:
        json.dump(data,outfile, indent=2)

    print(bcolors.OKGREEN+"Settings done."+bcolors.ENDC)
    print(f"\n{bcolors.OKGREEN}Set up crontab to run {bcolors.ENDC}"
        f"{bcolors.WARNING+cwd}main.sh{bcolors.ENDC}")
    print(f"{bcolors.OKGREEN}\nDon't forget to add tables to schema"
        f"/user you input here (templates in directory tbl_templates)."
        f"{bcolors.ENDC}")
    print(f"{bcolors.OKGREEN}\nYou can change the settings by running "
```

```
50.         f"this program again.{bcolors.ENDC}")
51.     print(f"{bcolors.OKGREEN}\nYou can run predict.py once the "
52.         f"database is full.{bcolors.ENDC}")
```

Code 16: Settings main script

```
1.  def set_storage(data):
2.      data["files"] = {}
3.      defaultFiles = {
4.          "model_storage" : "model_storage"
5.      }
6.      txt = input(f'Input directory to store the model ["'
7.                  f'{bcolors.WARNING+cwd+defaultFiles["model_storage"]
    +bcolors.ENDC}"]:')
8.      if txt is '':
9.          if not os.path.isdir(defaultFiles["model_storage"]):
10.             print("Directory not found, creating directory")
11.             os.makedirs(defaultFiles["model_storage"])
12.             print(bcolors.OKGREEN+"Directory created."+bcolors.ENDC)
13.                                 data["files"]["model_storage"]    =
    defaultFiles["model_storage"]
14.     else:
15.         if os.path.isdir(txt):
16.             data["files"]["model_storage"] = txt
17.         else:
18.                 prompt = query_yes_no("Directory not found, create
    it? :")
19.             if prompt:
20.                 os.makedirs(txt)
21.                 data["files"]["model_storage"] = txt
22.                                     print(bcolors.OKGREEN+"Directory
    created."+bcolors.ENDC)
23.             else:
24.                 sys.exit(f"{bcolors.FAIL}Couldn't continue, "
25.                 f"stopping the settings.{bcolors.ENDC}")
26.     return data
```

Code 17: Function to set model storage/folder

```
def set_db(data):
    # Procedure for database connections
    data["db"] = {}
    print(f"\n{bcolors.HEADER}Setting config for connecting to database"
        f" (get tablespace size, write and update storage data, etc.)"
        f"{bcolors.ENDC}")
    print(f"{bcolors.HEADER}Make sure that the user you insert here "
        f"have privilege to{bcolors.ENDC}")
    print(f"{bcolors.OKBLUE}CREATE SESSION, CREATE TABLE, CREATE VIEW, "
        f"CREATE PROCEDURE, CREATE SEQUENCE, CREATE TRIGGER{bcolors.ENDC}")
    print(f"{bcolors.OKBLUE}SELECT ON dba_free_space, SELECT ON "
        f"dba_data_files{bcolors.ENDC}")
    print(f"{bcolors.OKBLUE}[Optional] UNLIMITED TABLESPACE (you can "
        f"assign quota instead, but the data mined might be big and will "
        f"cause errors.){bcolors.ENDC}")
    data["db"]["hostname"] = input(f"Input hostname ["
        f"{bcolors.WARNING+'localhost'+bcolors.ENDC}]:") or "localhost"
    data["db"]["port"] = input(f"Input port ["
        f"{bcolors.WARNING+'1521'+bcolors.ENDC}]:") or "1521"
    data["db"]["service_name"] = input("Input service name :")
    data["db"]["tblspace"] = input("Input tablespace to predict :")
    data["master"] = Fernet.generate_key()
    kr.psDB(data["master"], "db", data)
    data["master"] = base64.b64encode(data["master"]).decode("utf-8")
    return data
```

Code 18: Function to set database configuration

```
def psDB(self, pwd, tipe, data):
    print(bcolors.HEADER+"\n\nSetting your oracle user and "\
        "password"+bcolors.ENDC)
    print(bcolors.HEADER+"Oracle user and password will be saved "\
        "and encrypted.\n"+bcolors.ENDC)
    pwd = self.chk_if_bytes(pwd)
    prompt_crd = query_yes_no(bcolors.HEADER+"Set new credential?"\
        +bcolors.ENDC, "yes")
    if not prompt_crd:
        cwd = os.path.dirname(os.path.realpath(__file__))
        filename = cwd+"/settings_storage/cred.json"
        if os.path.isfile(filename):
            print(bcolors.OKGREEN+"Credential found, continuing."\
                +bcolors.ENDC)
        else:
            print(bcolors.WARNING+"Credential not found, making "\
                "new one."+bcolors.ENDC)
            self.crd.setCred(tipe)
            print(bcolors.OKGREEN+"Credential created."\
                +bcolors.ENDC)
    else:
        self.crd.setCred(tipe)
        print(bcolors.OKGREEN+"Credential created."+bcolors.ENDC)

    self.file_kr.keyring_key = pwd
    while True:
```

```
27.         uname = input("Insert your username :")
28.         paswd = getpass.getpass(prompt="Insert your password:")
29.         if chk_con(data, uname, paswd):
30.                                     print(bcolors.OKGREEN+"Connection
    successful"+bcolors.ENDC)
31.             break
32.         else:
33.                                 print(bcolors.FAIL+"Cannot  connect  to
    database"+bcolors.ENDC)
34.       uname  =  self.chk_if_bytes(base64.b64encode(uname.encode("utf-
    8")))
35.     self.file_kr.set_password(self.crd.getSID(tipe),
36.                               self.crd.getUKEY(tipe), uname)
37.     self.file_kr.set_password(self.crd.getSID(tipe),
38.                               self.file_kr.get_password(
39.                                   self.crd.getSID(tipe),
40.                                   self.crd.getUKEY(tipe)),
41.                               paswd)
42.     del uname
43.     del paswd
```

Code 19: Function to set database user configuration

```
def psAlert(self,pwd,tipe):
    print(bcolors.HEADER+"\nSetting Alert"+bcolors.ENDC)
    print(bcolors.HEADER+"self alert will use email."
        +bcolors.ENDC)
    print(bcolors.HEADER+"You will enter your sender email and "\
        "password, and recipient email(email to recieve alert)."
        +bcolors.ENDC)
    print(bcolors.HEADER+"Sender email and password will be "\
        "encrypted and saved."+bcolors.ENDC)
    print(bcolors.WARNING+'Make sure you turn on "Less Secure '\
        '"App Access" in https://myaccount.google.com/security '\
        "for sender email.\n"+bcolors.ENDC)
    pwd = self.chk_if_bytes(pwd)
    cwd = os.path.dirname(os.path.realpath(__file__))
    filename = cwd+"/settings_storage/cred.json"
    if os.path.isfile(filename):
        print(bcolors.OKGREEN+"Credential found, modifying."
            +bcolors.ENDC)
        self.crd.setCred(tipe)
    else:
        print(bcolors.WARNING+"Credential not found."+bcolors.ENDC)
        sys.exit(bcolors.FAIL+"Couldn't continue, stopping the "\
            "settings, please check cred.json file in "\
            "settings_storage."+bcolors.ENDC)

    self.file_kr.keyring_key = pwd
    while True:
        send = input("Insert sender email:")
        if check_email(send):
            break
    send = self.chk_if_bytes(base64.b64encode(send.encode("utf-8")))
    self.file_kr.set_password(self.crd.getSID(tipe),
                              self.crd.getUKEY(tipe), send)
    self.file_kr.set_password(self.crd.getSID(tipe),
                              self.file_kr.get_password(
                                  self.crd.getSID(tipe),
                                  self.crd.getUKEY(tipe)),
                              getpass.getpass(prompt="Insert "\
                                              "sender password:"))
```

Code 20: Function to set alert user email

```
def set_thresh(data):
    print(bcolors.HEADER+"Insert size threshold."+bcolors.ENDC)
    print(bcolors.HEADER+"This size threshold (tablespace size) will "\
        "be used for alerting and the size when the tablespace is "\
        "considered 'Full'."+bcolors.ENDC)
    print(bcolors.HEADER+"You can use byte size (TB/GB/MB/B) or "\
        "precentages (%)."+bcolors.ENDC)
    print(bcolors.HEADER+"The interger part is the free space remaining."\
        " (i.e. 50MB means 50MB of free space remaining, 10% means 10% "\
        "free space remaining from max size.)"+bcolors.ENDC)
    while True:
        size = input("Insert the threshold size :").upper().replace(" ","")
        satuan = ["B","%"]
        if size[-1] in satuan:
            data["threshold_size"] = size
            break
        else:
            print("Size is invalid, try again")
    return data
```

Code 21: Function to set threshold size

```
def set_alert(data):
    if query_yes_no("Set up alerting module?"):
        data["alert"] = {}
        data["alert"]["alert_on"] = True
        kr.psAlert(base64.b64decode(data["master"]), "alert")
        while True:
            rec = input("Insert recipient email:")
            if check_email(rec):
                data["alert"]["recipient"] = rec
                break
        print(bcolors.OKGREEN+"Setting alert successfull."+bcolors.ENDC)
    else:
        data["alert"] = {}
        data["alert"]["alert_on"] = False
```

Code 22: Function to set alert module

```
def chng_conf(data):
    while True:
        os.system('clear')
        onoff=bcolors.FAIL+'OFF'+bcolors.ENDC if data['tracking']\
                else bcolors.OKGREEN+'ON'+bcolors.ENDC
        menus = [
            {"Change Model Storage": chng_model},
            {"Change Database": chng_db},
            {"Change Threshold Size": chng_thresh},
            {"Change Alert": chng_alert},
            {f"Turn tracking {onoff}": turn_tracking},
            {"Exit": exit}
        ]
        print("Enter the number of menu you want to change:")
        for item in menus:
            print("[" + str(menus.index(item)) + "] " + list(item.keys())[0])
        choice = input(">> ")
        try:
            if int(choice) < 0 : raise ValueError
            if int(choice)==5:
                break
            # Call the matching function
            data = list(menus[int(choice)].values())[0](data)
        except (ValueError, IndexError):
            pass

    with open(filename,'w') as outfile:
        json.dump(data,outfile, indent=2)
```

Code 23: main menu for settings after administrator run the settings one and configuration file saved.

```
def chng_model(data):
    txt =input(f'Input directory to store the model ["'\
                f'{bcolors.WARNING}'\
                +data["files"]["model_storage"]\
                +f'{bcolors.ENDC}"]:') or data["files"]["model_storage"]
    if txt is '':
        if not os.path.isdir(cwd+data["files"]["model_storage"]):
            print("Directory not found, creating directory")
            os.makedirs(cwd+data["files"]["model_storage"])
            print(bcolors.OKGREEN+"Directory created."+bcolors.ENDC)
    else:
        if os.path.isdir(cwd+txt):
            data["files"]["model_storage"] = txt
        else:
            prompt = query_yes_no("Directory not found, create it? :")
            if prompt:
                os.makedirs(cwd+txt)
                data["files"]["model_storage"] = txt
                print(bcolors.OKGREEN+"Directory created."+bcolors.ENDC)
            else:
                print("Using previous data instead.")
                if not os.path.isdir(cwd+data["files"]["model_storage"]):
                    print("Directory not found, creating directory")
                    os.makedirs(cwd+data["files"]["model_storage"])
```

```
                    print(bcolors.OKGREEN+"Directory created."\
                                +bcolors.ENDC)
    input("Press [ENTER] to continue...")
    return data
```

Code 24: Function to change model storage/folder

```
def chng_db(data):
    menus = [
        {"Change Hostname": chng_hostname},
        {"Change Port": chng_port},
        {"Change Service Name": chng_service},
        {"Change User": chng_user},
        {"Change Tablespace": chng_tblspc},
        {"Get back to main menu": exit}
    ]
    while True:
        os.system('clear')
        print("Enter the number of menu you want to change:")
        for item in menus:
            print("(" + str(menus.index(item)) + ") " + list(item.keys())[0])
        choice = input(">>>> ")
        try:
            if int(choice) < 0 : raise ValueError
            if int(choice)==5:
                break
            # Call the matching function
            data = list(menus[int(choice)].values())[0](data)
        except (ValueError, IndexError):
            pass
    return data
```

Code 25: Function to change database settings

```
def chng_hostname(data):
    data["db"]["hostname"] = input(f"Input hostname [{bcolors.WARNING}"
                                   +data['db']['hostname']
                                   +bcolors.ENDC
                                   +"] : ").strip()\
                                    or data['db']['hostname']
    return data

def chng_port(data):
    data["db"]["port"] = input(f"Input port [{bcolors.WARNING}"\
                                +data['db']['port']+bcolors.ENDC
                                +"] : ").strip() or data['db']['port']
    return data

def chng_service(data):
    data["db"]["service_name"] = input(f"Input hostname ["\
                                       +bcolors.WARNING
                                       +data['db']['service_name']
                                       +bcolors.ENDC
                                       +"}] : ").strip()\
                                       or data['db']['service_name']
    return data

def chng_tblspc(data):
    data["db"]["tblspace"] = input(f"Input tablespace [{bcolors.WARNING}"\
                                   +data['db']['tblspace']+bcolors.ENDC\
                                   +"]").strip() or data['db']['tblspace']
    return data
```

Code 26: Function to change hostname, port, service, or tablespace

```
1.  def chng_user(data):
2.      pwd = base64.b64decode(data["master"]).decode("utf-8")
3.      kr.modDB(pwd, "db", data)
4.      input("Press [ENTER] to continue...")
5.      return data
6.
7.  def modDB(self, pwd, tipe, data):
8.      self.file_kr.keyring_key = self.chk_if_bytes(pwd)
9.      print(bcolors.HEADER+"\nChanging your oracle user and password"\
10.             +bcolors.ENDC)
11.     uname = self.file_kr.get_password(self.crd.getSID(tipe),
12.                                       self.crd.getUKEY(tipe))
13.     self.file_kr.delete_password(self.crd.getSID(tipe), uname)
14.     print(bcolors.OKGREEN+"Credential created."+bcolors.ENDC)
15.     while True:
16.         uname = input("Insert your username :")
17.         paswd = getpass.getpass(prompt="Insert your password:")
18.         if chk_con(data, uname, paswd):
19.             print(bcolors.OKGREEN+"Connection successful"
20.                   +bcolors.ENDC)
21.             break
22.         else:
23.             print(bcolors.FAIL+"Cannot connect to database"
24.                   +bcolors.ENDC)
25.     uname = self.chk_if_bytes(base64.b64encode(
26.                                       uname.encode("utf-8")
27.                                       )
28.                               )
29.     self.file_kr.set_password(self.crd.getSID(tipe),
30.                               self.crd.getUKEY(tipe), uname)
31.     self.file_kr.set_password(self.crd.getSID(tipe),
32.                               self.file_kr.get_password(
33.                                       self.crd.getSID(tipe),
34.                                       self.crd.getUKEY(tipe)),
35.                               paswd)
36.     print(bcolors.OKGREEN+"Changes complete"+bcolors.ENDC)
37.     del uname
38.     del paswd
```

Code 27: Function to change database user

```
def chng_thresh(data):
    print(bcolors.HEADER+"Insert size threshold."+bcolors.ENDC)
    print(bcolors.HEADER+"This size threshold (tablespace size) will be"\
        " used for alerting and the size when the tablespace is "\
        "considered 'Full'."+bcolors.ENDC)
    print(bcolors.HEADER+"You can use byte size (TB/GB/MB/B) or "\
        "precentages (%)."+bcolors.ENDC)
    print(bcolors.HEADER+"The interger part is the free space "\
        "remaining. (i.e. 50MB means 50MB of free space remaining, "\
        "10% means 10% free space remaining from max size.)"+bcolors.ENDC)
    while True:
        size = input(f"Insert the threshold size [{bcolors.WARNING}"
                    +data['threshold_size']+bcolors.ENDC\
                    +"]:").upper().replace(" ","") \
                    or data["threshold_size"]
        satuan = ["B","%"]
        if size[-1] in satuan:
            data["threshold_size"] = size
            break
        else:
            print("Size is invalid, try again")
    input("Press [ENTER] to continue...")
    return data
```

Code 28: Function tho change threshold size

```
def chng_alert(data):
    while True:
        onoff = bcolors.FAIL+'OFF'+bcolors.ENDC if data['alert']['alert_on']\
                else bcolors.OKGREEN+'ON'+bcolors.ENDC
        menus = [
            {f"Turn Alert {onoff} ": turn_alert},
            {"Change Sender": chng_send},
            {"Change Receiver": chng_rec},
            {"Get back to main menu": exit}
        ]
        os.system('clear')
        print("Enter the number of menu you want to change:")
        for item in menus:
            print("(" + str(menus.index(item)) + ") "
                + list(item.keys())[0])
        choice = input(">>>> ")
        try:
            if int(choice) < 0 : raise ValueError
            # Call the matching function
            if int(choice)==3:
                break
            data = list(menus[int(choice)].values())[0](data)
        except (ValueError, IndexError):
            pass
    return data
```

Code 29: Function for alert menu

```
def turn_alert(data):
    if data['alert']['alert_on']:
        data['alert']['alert_on'] = False
    else:
        data['alert']['alert_on'] = True
    return data

def chng_rec(data):
    while True:
        rec = input("Insert recipient email:")
        if check_email(rec):
            data["alert"]["recipient"] = rec
            break
    input("Press [ENTER] to continue")
    return data
```

Code 30: Function to turn alert on/off and recipient email

```
def chng_send(data):
    pwd = base64.b64decode(data["master"]).decode("utf-8")
    kr.modAlert(pwd, "alert")
    input("Press [ENTER] to continue")
    return data

def modAlert(self, pwd, tipe):
    self.file_kr.keyring_key = self.chk_if_bytes(pwd)
    print(bcolors.HEADER+"\nChanging your sender email and password"\
          +bcolors.ENDC)
    email = self.get_old_email()
    self.file_kr.delete_password(self.crd.getSID(tipe), email)
    while True:
        send = input("Insert sender email:")
        if check_email(send):
            break
    send = self.chk_if_bytes(base64.b64encode(send.encode("utf-8")))
    self.file_kr.set_password(self.crd.getSID(tipe),
                              self.crd.getUKEY(tipe), send)
    self.file_kr.set_password(self.crd.getSID(tipe),
                              self.file_kr.get_password(
                                      self.crd.getSID(tipe),
                                      self.crd.getUKEY(tipe)),
                              getpass.getpass(prompt="Insert "\
                                                     "your password:"))
    print(bcolors.OKGREEN+"Changes complete"+bcolors.ENDC)
```

Code 31: Function to change sender email & password

```
def turn_tracking(data):
    if data['tracking']:
        data['tracking'] = False
    else:
        data['tracking'] = True
    return data
```

Code 32: Function to turn tracking target tablespace on/off

ii. Mining Process

```
if __name__ == "__main__":
    app = main()
    if app.config["tracking"]:
        app.start()
        app.close()
    else:
        print("Terminating.")
```

Code 33: Code to invoke main program

```
def start(self):
    """ Function to start mining process """
    tblspc_data = self.gtdata.start() # start to save data
    """ Check if free space is below threshold"""
    if tblspc_data[0]:
        """Check if the alert module is on"""
        if self.config["alert"]["alert_on"]:
            cur_data = tblspc_data[1] # get current space
            if len(cur_data)>0:
                self.alert(cur_data) #alert admin
        if len(tblspc_data) == 3:
            self.train(True) # Train using new data (repeat)
        else:
            self.train(False) # Train using new data
```

Code 34: Main function that control the whole mining process, invoking alert and train process

```
def start(self):
    """ Function to save data to database, return List"""
    data = self.ftch_data() #get size data from tablespace
    if data is not None:
        log_check = self.get_log()
        if log_check:
            """ Check if log start_at is inside the tbldate"""
            if len(self.sds.getIdDate(dateutil.parser.parse(
                    log_check[self.tablespace]["start_at"])))==0:
                if self.chk_total():
                    self.new_tblspc_log(with_root=False)
                else:
                    self.new_tblspc_log()
        id_date = self.sds.getIdDate(self.tglwaktu)
        if len(id_date) > 0 :
            id_date = id_date[0][0]
        else:
            self.sds.ins_date(self.tglwaktu)
            id_date = self.sds.getIdDate(self.tglwaktu)[0][0]
        flag = self.compare_size(data[1], data[2])
        """ Check if it can compare size"""
        if flag:
            """ Cek if the size more or less than threshold"""
            if flag == "<":
                return self.above_thresh(data)
            else:
                """If there's already data in table"""
                if self.chk_total():
                    if len(self.chk_out_null()) == 0:
                        """ If no data found with null id_output"""
                        """ use id_output from last input row"""
                        return self.below_thresh_already_full(data)
                    else:
                        """ If data found with null id_output"""
                        """ make new output in id_output"""
                        id_out = self.sds.cekTblOut(id_date)
                        if len(id_out)==0:
                            """ Insert new output inside output table then ""
                            """ check if it succeed."""
                            if self.sds.insOut(id_date):
                                id_out = self.sds.cekTblOut(id_date)
                                if len(id_out)>0:
                                    self.below_thresh_new(data,id_out)
                                else:
                                    print("Fail to add output")
                        else:
                            """ jika ditemukan id_output dengan hari ini"""
                            self.below_thresh_new(data, id_out)
                else:
                    print("No data found while the space is full.")
                    print("Inserting new data.")
                    """ Masukkan output baru lalu cek keberhasilan"""
                    if self.sds.insOut(id_date):
                        id_out = self.sds.cekTblOut(id_date)
                        if len(id_out)>0:
                            self.below_thresh_no_data(data, id_out)
```

```
                    else:
                        print("Fail to add output")
                return [True, data]
    else:
        print("Cannot find tablespace.")
        return [False]
```

Code 35 : Function to insert raw data to dataset schema

```
def ftch_data(self):
    """ Function to fetch data of tablespace size, return 1D List"""
    sql = """
        select b.tablespace_name, tbs_size SizeMb,
        a.free_space FreeMb
        from  (select tablespace_name, round(sum(bytes)/1024/1024 ,2)
            as free_space
            from dba_free_space
            group by tablespace_name) a,
          (select tablespace_name, sum(bytes)/1024/1024 as tbs_size
           from dba_data_files
           group by tablespace_name) b
        where a.tablespace_name(+)=b.tablespace_name"""
    c = self.orcl.execute(sql, {})
    datas = c.fetchall()
    ftch = None
    for row in datas:
  """ Check data with tablespace name matched with config file """
        if row[0] == self.tablespace.upper():
            ftch = row
    return ftch
```

Code 36: Function to get current tablespace size

```
1.  def get_log(self):
2.  """Function to get log (start_at & root_start), return
    List/bool"""
3.      cwd = os.path.dirname(os.path.realpath(__file__))+"/../"
4.      filename = cwd+"settings_storage/log.json"
5.      if os.path.isfile(filename):
6.          with open(filename) as json_data_file:
7.              data = json.load(json_data_file)
8.          if self.tablespace in data:
9.              return data
10.         else:
11.             return False
12.     else:
13.         return False
```

Code 37: Function read the log file that record cycle start date and date when the system start tracking.

```
1.  def chk_total(self):
2.      """Function to fetch total row based on id_tblspc,
    return Bool/list"""
3.      sql = """
4.          select count(*)
5.          from  tblinput
6.          where id_tblspc = :id_tblspc
7.          group by id_tblspc
8.      """
9.      id_tblspc = self.sds.getIdTblspc(self.tablespace)
10.     if len(id_tblspc)>0:
11.         id_tblspc=id_tblspc[0][0]
12.         c = self.orcl.execute(sql, {"id_tblspc":
    id_tblspc})
13.         datas = c.fetchall()
14.         if len(datas)>0:
15.             return datas
16.         else:
17.             return False
18.     else:
19.         return False
```

Code 38: Function to get total dataset row from tablespace.

```
1. def new_tblspc_log(self, with_root = True):
2.     """Function untuk membuat log"""
3.     cwd = os.path.dirname(os.path.realpath(__file__))+"/../"
4.     filename = cwd+"settings_storage/log.json"
5.     if os.path.isfile(filename):
6.         with open(filename) as json_data_file:
7.             log = json.load(json_data_file)
8.     else:
9.         log = {}
10.    if self.tablespace not in log:
11.        log[self.tablespace]={}
12.    log[self.tablespace]["start_at"] = self.tglwaktu.isoformat()
13.    if with_root:
14.        log[self.tablespace]["root_start"] =
   self.tglwaktu.isoformat()
15.    with open(filename,'w') as outfile:
16.        json.dump(log,outfile, indent=2)
```

Code 39: Function new_tblspc_log() check if log file exists, it create new one if it doesn't exist. If with_root size is true, save the current date as root_start.

```
1. def compare_size(self, max_size, free):
2.     """Function untuk membandingkan ukuran, return String"""
3.     bigSize = [
4.         "kb",
5.         "mb",
6.         "gb",
7.         "tb"
8.     ]
9.     if free is None:
10.        free = 0
11.    """cek jika threshold persen"""
12.    if self.thresh[-1:] == "%":
13.        pct = free / max_size * 100
14.        return self.bandingkan(float(self.thresh[:-1]), pct)
15.    else:
16.        if self.thresh[-2:].lower() in bigSize:
17.            byte_thresh = self.get_bytes(self.thresh[:-2], \
18.                                         self.thresh[-2:])
19.            return self.bandingkan(byte_thresh, \
20.                                   self.get_bytes(free,"mb"))
21.        elif self.thresh[-1:].lower() == "b":
22.            byte_thresh = self.thresh[:-1]
23.            return self.bandingkan(byte_thresh, \
24.                                   self.get_bytes(free,"mb"))
25.        else:
26.            print("Error. Check suffix.")
27.            return False
28.
29. def bandingkan(self, size1, size2):
30.    """Function untuk membandingkan ukuran, return String"""
31.    size1= int(size1)
32.    size2= int(size2)
33.    if size1>=size2:
34.        return ">="
35.    elif size1<size2:
36.        return "<"
37.
38. def get_bytes(self, size, suffix):
39.    """Function untuk convert ke bytes, return int"""
40.    size = int(float(size))
41.    suffix = suffix.lower()
42.    if suffix == 'kb' or suffix == 'kib':
43.        return size * 1024
44.    elif suffix == 'mb' or suffix == 'mib':
45.        return size * 1024 ** 2
46.    elif suffix == 'gb' or suffix == 'gib':
47.        return size * 1024 ** 3
48.    elif suffix == 'tb' or suffix == 'tib':
49.        return size * 1024 ** 4
50.    return False
```

Code 40: Function to compare the data size with threshold

```
def above_thresh(self, data):
    """ Check if data already exists in schema"""
    if self.chk_total():
        last_in = self.sds.get_last_input(
                    self.sds.getIdTblspc(self.tablespace)[0][0]
                    ) # get id_output from last input
        """ check if id_output is null"""
        if not last_in[0][0] is None:
            """ if not null, update log """
            self.mod_log()
    log = self.get_log()
    """ check if log doesn't exists """
    if not log:
        """ create new """
        self.new_tblspc_log()
        log = self.get_log()
    data_to_save = {}
    data_to_save["tanggalwaktu"] = self.tglwaktu
    data_to_save["nama_tblspc"] = self.tablespace
    data_to_save["max_size"] = data[1]
    data_to_save["free_space"] = data[2]
    data_to_save["start_at"] = dateutil.parser.parse(
                                log[self.tablespace]["start_at"])
    data_to_save["root_start"] = dateutil.parser.parse(
                                log[self.tablespace]["root_start"])
    self.sds.ins_data(data_to_save) #Insert the data
    return [False]
```

Code 41: Function that gets called when the current data size is above threshold

```
def below_thresh_already_full(self, data):
    last_in = self.sds.get_last_input(
                self.sds.getIdTblspc(
                        self.tablespace)[0][0])
    log = self.get_log()
    data_to_save = {}
    data_to_save["tanggalwaktu"] = self.tglwaktu
    data_to_save["nama_tblspc"] = self.tablespace
    data_to_save["max_size"] = data[1]
    data_to_save["free_space"] = data[2]
    data_to_save["id_output"] = last_in[0][0]
    data_to_save["start_at"] = dateutil.parser.parse(
                                log[self.tablespace]["start_at"]
                                )
    data_to_save["root_start"] = dateutil.parser.parse(
                                log[self.tablespace]["root_start"]
                                )
    self.sds.ins_data(data_to_save, True)
    return [True, data, True]
```

Code 42: Function below_thresh_alread_full() get called when it can't find any previous data that has empty id_output (this means this cycle is already full and it uses id_output of last row inserted in dataset table of schema).

```
def insOut(self, id_date):
    sql = """
        INSERT INTO TBLOUTPUT
        VALUES(NULL, :id_date)
    """
    try:
        c = self.orclobj.execute(sql,{'id_date' : id_date}, True)
        return True
    except:
        return False
```

Code 43: Function for inserting new output in tbloutput.

```
def below_thresh_no_data(self, data, id_out):
    log = self.get_log()
    if not log:
        self.new_tblspc_log()
        log = self.get_log()
    data_to_save = {}
    data_to_save["tanggalwaktu"] = self.tglwaktu
    data_to_save["nama_tblspc"] = self.tablespace
    data_to_save["max_size"] = data[1]
    data_to_save["free_space"] = data[2]
    data_to_save["id_output"] = id_out[0][0]
    data_to_save["start_at"] = dateutil.parser.parse(
                                log[self.tablespace]
                                ["start_at"])
    data_to_save["root_start"] = dateutil.parser.parse(
                                log[self.tablespace]
                                ["root_start"])
    # Insert data with id_output
    self.sds.ins_data(data_to_save, True)
```

Code 44: Function that gets called if there's no data in dataset schema and tablespace already full

```
def below_thresh_new(self, data, id_out):
    log = self.get_log()
    data_to_save = {}
    data_to_save["tanggalwaktu"] = self.tglwaktu
    data_to_save["nama_tblspc"] = self.tablespace
    data_to_save["max_size"] = data[1]
    data_to_save["free_space"] = data[2]
    data_to_save["id_output"] = id_out[0][0]
    data_to_save["start_at"] = dateutil.parser.parse(\
                                log[self.tablespace]["start_at"])
    data_to_save["root_start"] = dateutil.parser.parse(\
                                log[self.tablespace]["root_start"])
    # Insert data with id_output
    self.sds.ins_data(data_to_save, True)
    # update all previous data that have null id_output
    self.sds.updateAllNull(id_out[0][0])
```

Code 45: Function below_thresh_new() get called when it is the first time of the day it fetch data and cycle the database is full.

```
def alert(self, cur_data):
    """ Function untuk alert admin """
    email = mail()
    data = {
        "recipient" : self.config["alert"]["recipient"],
        "thresh"    : self.config["threshold_size"]
    }
    email.send(cur_data, self.tgl, data)
```

Code 46: Function to alert admin

```
def send(self, cur_data, tgl, data):
    user = self.get_email()
    sent_from = "Forecast Alert System"
    to = data["recipient"]
    message = MIMEMultipart("alternative")
    message["Subject"] = "Alert for forecast (Tablespace full)"
    message["From"] = sent_from
    message["To"] = to
    email_text ='''
    Hi! Today's date is {0}.
    This is an alert to let you know that your tracked tablespace'''\
    ''' is full!\nCurrently your tablespace size is {1}MB with the '''\
    '''maximum size of {2}MB, less than the current threshold which '''\
    '''is {3}.\n Please add datafile to the tablespace.
    Also, you can turn off the tracking and alert via settings.


    Have a nice day!'''
    email_text_html = '''
    <html>
        <body>
            Hi! Today's date is <b>{0}</b>.<br>
            This is an alert to let you know that your tracked '''\
            '''tablespace is <strong>full</strong>!<br>
            Currently your tablespace size is <mark><b>{1}MB</b>'''\
            '''</mark> with the maximum size of <b>{2}MB</b>, less '''\
            '''than the current threshold which is <mark><b>{3}</b>'''\
            '''</mark>.<br>
            Please <i>add datafile</i> to the tablespace.<br>
            Also, you can <i>turn off</i> the <i>tracking</i> and <i>'''\
            '''alert</i> via <i>settings</i>.<br>
            <br><br>
            Have a nice day!
        </body>
    </html>
    '''
    email_text = email_text.format(tgl.strftime("%d/%m/%Y"),
                                   cur_data[2], cur_data[1],
                                   data["thresh"])
    email_text_html = email_text_html.format(tgl.strftime("%d/%m/%Y"),
                                             cur_data[2], cur_data[1],
                                             data["thresh"])
    part = MIMEText(email_text,"plain")
    part2 = MIMEText(email_text_html,"html")
```

```
45.     message.attach(part)
46.     message.attach(part2)
47.     #email send request
48.     try:
49.         server = smtplib.SMTP_SSL('smtp.gmail.com', 465)
50.         server.ehlo()
51.         server.login(user, self.get_pass())
52.         server.sendmail(user, to, message.as_string())
53.         server.close()
54.         print ('Email sent!')
55.     except Exception as e:
56.         print(e)
57.         print ('Something went wrong...')
```

Code 47: Function to send email

```
def start(self, diff, append_last = False):
    if len(self.getDsLists())==0:
        self.del_log()
    if not append_last:
        for dt in diff:
            self.w_u_log_train()
            ds_ke = self.r_log()["ds_ke"]
            tglwaktu = dt
            sql = """
            SELECT A.*,E.TANGGAL AS START_DATE, B.TANGGAL AS IN_DATE, """
            """D.TANGGAL AS OUT_DATE
            FROM TBLINPUT A
            INNER JOIN TBLDATE B ON A.ID_DATE = B.ID_DATE
            INNER JOIN TBLOUTPUT C ON A.ID_OUTPUT = C.ID_OUTPUT
            INNER JOIN TBLDATE D ON C.ID_DATE = D.ID_DATE
            INNER JOIN TBLDATE E ON A.START_AT = E.ID_DATE
            WHERE A.ID_TBLSPC = :id_tblspc
            AND E.TANGGAL = TRUNC(:tglwaktu)
            ORDER BY A.ID_INPUT
            """
            c = self.orcl.execute(sql,{"id_tblspc":self.id_tblspc, \
                                       "tglwaktu": tglwaktu})
            data = c.fetchall()
            data2 = self.process_data(data, dt, ds_ke)
            header = [
                "DATA_KE",
                "DS_KE",
                "TIPE_TGL",
                "MINGGU_KE",
                "DAYOFWEEK" ,
                "INC_SIZE" ,
                "FREE_SPACE" ,
                "ACTUAL_OUT" ]
            idx_row = 0
            for row in data:
                not_last = True
                jumlah = self.get_jumlah(row[-3])
                if os.path.isfile(self.fp+"/datasets/"+str(row[2])+"."\
                                  +row[-3].isoformat()+'.csv'):
                    with open(self.fp+"/datasets/"+str(row[2])+"."\
                              +row[-3].isoformat()+'.csv', "r" ) \
                            as readFile:
                        reader = csv.reader(readFile)
                        lines = list(reader)

                        if len(lines) >= jumlah[0]+1:
                            not_last = False
                    if not_last:
                        with open(self.fp+"/datasets/"+str(row[2])+"."\
                                  +row[-3].isoformat()+'.csv','a')\
                                as writeFile:
                            writer = csv.writer(writeFile)
                            writer.writerow(data2[idx_row])
                        writeFile.close()
                else:
                    with open(self.fp+"/datasets/"+str(row[2])+"."\
                              +row[-3].isoformat()+'.csv','a') \
```

```
                            as writeFile:
                        writer = csv.writer(writeFile)
                        writer.writerow(header)
                        writer.writerow(data2[idx_row])
                    writeFile.close()
                idx_row = idx_row+1
    else:
        sql = """
        SELECT A.*,E.TANGGAL AS START_DATE, B.TANGGAL AS IN_DATE, """
        """D.TANGGAL AS OUT_DATE
        FROM TBLINPUT A
        INNER JOIN TBLDATE B ON A.ID_DATE = B.ID_DATE
        INNER JOIN TBLOUTPUT C ON A.ID_OUTPUT = C.ID_OUTPUT
        INNER JOIN TBLDATE D ON C.ID_DATE = D.ID_DATE
        INNER JOIN TBLDATE E ON A.START_AT = E.ID_DATE
        WHERE A.ID_TBLSPC = :id_tblspc
        AND E.TANGGAL = TRUNC(:tglwaktu)
        ORDER BY A.ID_INPUT DESC
        FETCH FIRST 1 ROWS ONLY
        """
        c = self.orcl.execute(sql,{"id_tblspc":self.id_tblspc, \
                                   "tglwaktu": diff})
        data = c.fetchall()
        ds_ke = self.r_log()["ds_ke"]
        data2 = self.process_data(data, dt, ds_ke)
        if os.path.isfile(self.fp+"/datasets/"+str(data[0][2])+"."\
                          +data[0][-3].isoformat()+'.csv'):
            with open(self.fp+"/datasets/"+str(data[0][2])+"."\
                      +data[0][-3].isoformat()+'.csv','a')\
                    as writeFile:
                writer = csv.writer(writeFile)
                writer.writerow(data2[0])
            writeFile.close()
```

Code 48: Function cDataset.start() task is to fetch all cycle start date from list that are given by main class process it and then convert it to CSV

```
1. def process_data(self, data, dt, ds_ke):
2.     ret_data=[]
3.     avrg = data[0][5]/(data[0][-1] - data[0][-2]).days
4.     print(str(dt)+" : Average data stored per day = "+str(avrg))
5.
6.     savedate = None
7.     for row in data:
8.         size_flag = self.compare_size(row[4], row[5])
9.         if size_flag:
10.            if size_flag == ">=":
11.                savedate = row[-2]
12.                break
13.    if savedate is None:
14.        savedate = data[-1][-2]
15.
16.    data_ke = 1
17.    flag=1
18.    for row in data:
19.        tmp = []
20.        tmp.append(data_ke)
21.        if flag%10==0:
22.            data_ke = data_ke+1
23.        flag = flag+1
24.
25.        tmp.append(ds_ke)
26.        tmp.append(row[5]) # current size
27.        sisa_hari = savedate - row[-2]
28.        if sisa_hari.days < 0:
29.            tmp.append(0)
30.        else:
31.            tmp.append(sisa_hari.days) # The actual output
32.        ret_data.append(tmp)
33.    return ret_data
```

Code 49: Function to process the data before storing it as CSV

```
def set_model(self, in_total):
    alph = 0.01
    self.model = Sequential()
    self.model.add(Dense(2*in_total,
                         input_dim=in_total))
    self.model.add(LeakyReLU(alph))
    self.model.add(Dense(int(in_total+(in_total/2))))
    self.model.add(LeakyReLU(alph))
    self.model.add(Dense(in_total))
    self.model.add(LeakyReLU(alph))
    self.model.add(Dense(1))
    self.model.add(LeakyReLU(alph))
    opt = Adam(lr = 0.001)
    self.model.compile(loss='mse', optimizer=opt, metrics=['mse'])
```

Code 50: This function set the neural network model

```
def train(self, repeat = False):
    """ Function untuk training NN """
    diff = self.get_last_dataset() # get start date that isn't in file
    if diff:
        if len(diff)>0:
            dset = cDataset(self.config)
            dset.start(diff,False) # make the dataset (csv file)
            dset.close()
            pwd = os.path.dirname(os.path.realpath(__file__))
            for dt in diff:
                """ Train untuk dataset terbaru yang belum ditraining """
                train = Train(dset.id_tblspc, dt, pwd, self.config["files"])
                reset = train.start()
                if reset:
                    print("restart training")
                    self.restart_train()
                    break
        print("Training done")
    else:
        if repeat:
            lst_ds = self.gtdata.get_all_dataset()[-1][0]
            dset = cDataset(self.config)
            dset.start(lst_ds,True) # add dataset row (file csv)
            dset.close()
            pwd = os.path.dirname(os.path.realpath(__file__))
            train = Train(dset.id_tblspc, lst_ds, pwd, self.config["files"])
            reset = train.start()
            if reset:
                print("restart training")
                self.restart_train()
                break
            print("Training done")
        else:
            print("Couldn't continue training")
```

Code 51: Function train in the main work as controller for the entire training process.

```
def start(self):
    in_total = 6
    train_x = self.dataset[:,0:in_total]
    train_y = self.dataset[:,in_total]
    test_x = self.test[:,0:in_total]
    test_y = self.test[:,in_total]
    if not os.path.isfile(self.pathdir+"/"\
            +self.model_storage+"/model.h5"):
        self.set_model(in_total)
    else:
        print("Using past model")
        self.model = None
        self.model = self.load_mdl(self.pathdir, \
                                   self.model_storage)
    history = self.model.fit(train_x, train_y, \
                validation_data=(test_x, test_y), \
                epochs=50, batch_size=10, verbose=0)
    _,train_mse = self.model.evaluate(train_x, train_y, \
                    verbose=1)
    _,test_mse = self.model.evaluate(test_x, test_y, \
                    verbose=1)
    print('Train: %.3f, Test: %.3f' % (train_mse, test_mse))
    with open("train_nn/loss.csv",'a') as writeFile:
        writer = csv.writer(writeFile)
        for row in history.history['loss']:
            writer.writerow([row])
    writeFile.close()
    with open("train_nn/val_loss.csv",'a') as writeFile:
        writer = csv.writer(writeFile)
        for row in history.history['val_loss']:
            writer.writerow([row])
    writeFile.close()
    predictions = self.model.predict(train_x)
    correct = 0
    false = 0
    one_day_diff = 0
    two_day_diff = 0
    more = 0
    for i in range(len(train_x)):
        if int(predictions[i][0]) != int(train_y[i]):
            if abs(int(predictions[i][0]) \
                - int(train_y[i])) >=3 :
                more = more + 1
            else:
                if abs(int(predictions[i][0])\
                    - int(train_y[i])) == 1:
                    one_day_diff = one_day_diff + 1
                else:
                    two_day_diff = two_day_diff + 1
            false = false+1
        else:
            correct = correct+1
    print(f'Correct = {correct} ; False = {false}')
    print(f'1 day diff = {one_day_diff}; 2 day diff = '\
            f'{two_day_diff}; more or equal than 3 day = '\
```

```
56.             f'{more}')
57.     acc = (correct+one_day_diff) / (correct+false) *100
58.     print(f'Accuracy (with 1 day tolerance): {acc}')
59.     self.save_mdl(self.pathdir, self.model_storage)
60.     return False
```

Code 52: This function trains the neural network. After training is done, save the model as h5 format (line 59).

(2) Predicting Process

```
def __init__(self, pathdir, config):
    self.pathdir = pathdir
    self.tblspc = config["db"]["tblspace"]
    self.model_storage = config["files"]["model_storage"]
    self.orcl = Oracle()
    file_kr = CryptFileKeyring()
    crd = cred()
    if os.path.isfile(pathdir+"/"+self.model_storage+"/model.h5"):
        tipe = "db"
        file_kr.keyring_key = base64.b64decode(config["master"])\
                                    .decode("utf-8")
        uname = self.chk_if_bytes(base64.b64decode(file_kr\
                                        .get_password(
                                            crd.getSID(tipe),
                                            crd.getUKEY(tipe)
                                        )
                                    )
                                )
        paswd = file_kr.get_password(crd.getSID(tipe), file_kr\
                                    .get_password(
                                        crd.getSID(tipe),
                                        crd.getUKEY(tipe)
                                    )
                                )
        if uname != None and paswd != None:
            while True:
                if chk_con(config, uname, paswd):
                    break
                else:
                    print("Cannot connect to database")
                    exit()
            res = self.orcl.connect(uname, paswd,
                                    hostname=config["db"]["hostname"],
                                    port=config["db"]["port"],
                                    servicename=config["db"]\
                                                ["service_name"])
        self.model = self.load_mdl(pathdir, self.model_storage)
        self.dt = datetime.now()
        self.input_size = 7
    else:
        print("Cannot find model, terminating.'")
        exit()
```

Code 53: Initializing class, check connection to database, save the required variable to run the predicting process, and load the neural network model.

```
1. def start(self):
2.     size_data = self.ftch_data()
3.     data = self.prep_data(size_data)
4.     predictions = self.model.predict(data)
5.     pred_res = abs(predictions[0][0])
6.     print(bcolors.OKGREEN+str(pred_res)+" day(s) until full"+""+bcolors.ENDC)
```

Code 54: Main function to predict data

```
1. def prep_data(self, data, dt, ds_ke):
2.     ret_data=[]
3.     avrg = data[0][5]/(data[0][-1] - data[0][-2]).days
4.     print(str(dt)+" : Average data stored per day = "+str(avrg))
5.
6.     savedate = None
7.     for row in data:
8.         size_flag = self.compare_size(row[4], row[5])
9.         if size_flag:
10.            if size_flag == ">=":
11.                savedate = row[-2]
12.                break
13.    if savedate is None:
14.        savedate = data[-1][-2]
15.
16.    data_ke = 1
17.    flag=1
18.    for row in data:
19.        tmp = []
20.        tmp.append(data_ke)
21.        if flag%10==0:
22.            data_ke = data_ke+1
23.        flag = flag+1
24.
25.        tmp.append(ds_ke)
26.        tmp.append(row[5]) # current size
27.        sisa_hari = savedate - row[-2]
28.        if sisa_hari.days < 0:
29.            tmp.append(0)
30.        else:
31.            tmp.append(sisa_hari.days) # The actual output
32.        ret_data.append(tmp)
33.    return ret_data
```

Code 55: Function to process the data before predicting

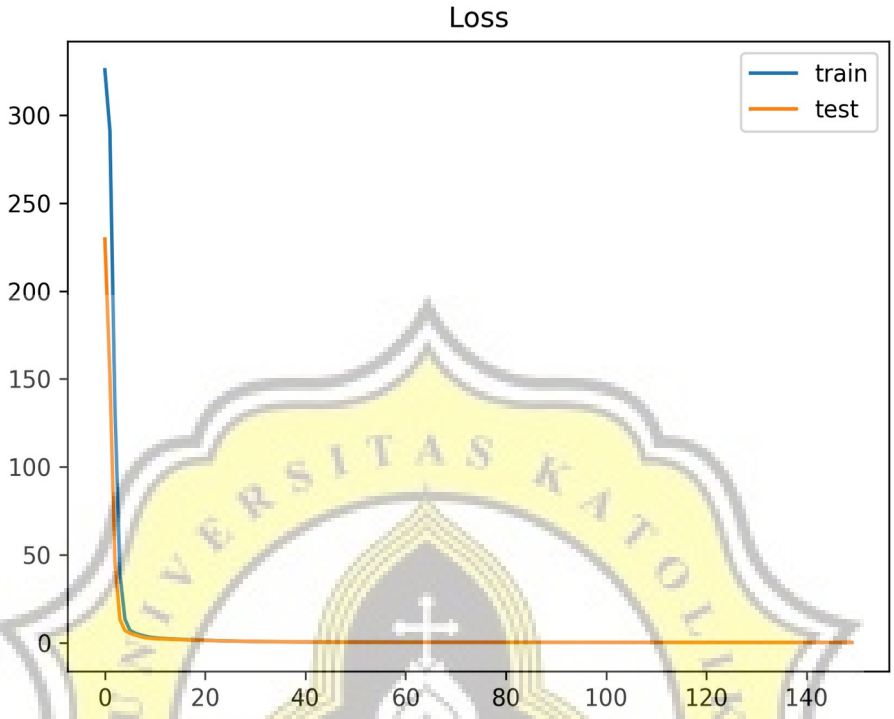

Illustration 6.1: Example plot graph of loss. Taken from tblspc1, train until cycle 1, sample 1

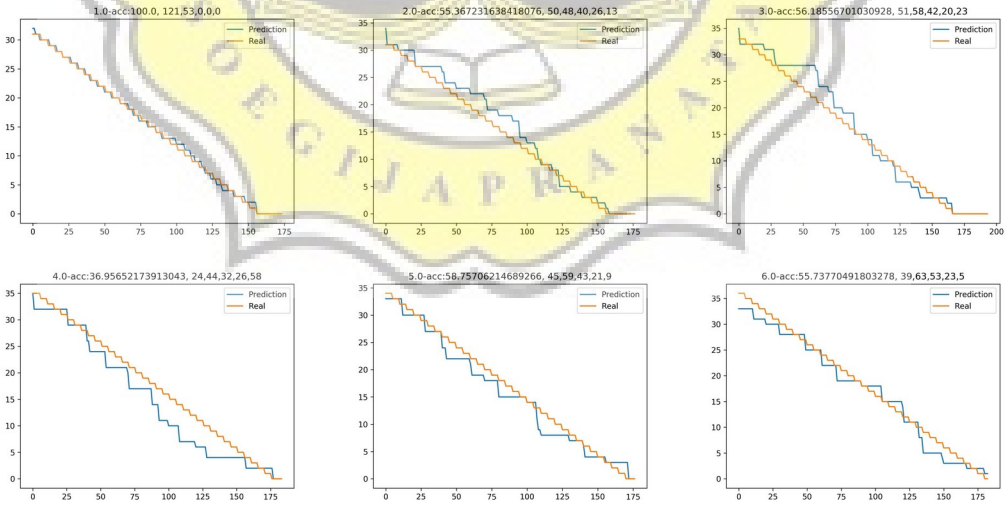

Illustration 6.2: Example of plot diagram of prediction and real value. Taken from tblspc1, train until cycle 1, sample 1

UNICHECK

Submission author:
16k10028 CHRISTOPHORUS REYHAN T. A.

Check ID:
15890108

Check date:
14.01.2020 08:47:37 GMT+0

Check type:
Doc vs Internet + Library

Report date:
15.01.2020 03:09:02 GMT+0

User ID:
29152

File name: 16.K1.0028_Christophorus Reyhan.doc

File ID: 20187121 Page count: 25 Word count: 11076 Character count: 66390 File size: 241.50 KB

5.49% Matches

Highest match: 1.47% with source https://stats.stackexchange.com/a/176905/98975?source=post_page-----ec68f76ffb9b----------...

5.38% Internet Matches	69	Page 27
0.25% Library matches	41	Page 27

1.97% Quotes

Quotes	8	Page 28

No references found

0% Exclusions

No exclusions found

Replacement

Character replacement	1

AM