ARTIFICAL INTELLIGENCE

PROJECT PART 1

REPORT

Areeba Sattar I200634

Maria Saeed I200836

Step 1:

In the 1st step, we read all the data from the files and store them in data frames.

```
file_loc = "C:/Users/DELL/Downloads/ai_proj/training_data/training_data/cluster_map/cluster_map"


cluster_map = pd.read_csv(file_loc, sep='\t', header=None, names=["region_hash","id_region"])
✓ 0.0s
```

First we have read the cluster and stored it in data frame.

Then we read order details.

```
file_loc = "C:/Users/DELL/Downloads/ai_proj/training_data/training_data/order_data/order_data_2016-01-01"
size = len(file_loc)
order_dets = pd.DataFrame(columns=["order_id","driver_id","passenger_id","start_region_hash","dest_region_hash","Price","Time"])
#=[None] *22
order_deets = pd.DataFrame(columns=["order_id","driver_id","passenger_id","start_region_hash","dest_region_hash","Price","Time"])

#
for i in range(1,10):
    #file_loc = file_loc.replace(file_loc[size - 1], str(i))
    last=str(i)
    file_loc=file_loc[:- 1]+last
    # print(file_loc)

    order_dets = pd.read_csv(file_loc, sep='\t', header=None, names=["order_id","driver_id","passenger_id","start_region_hash","dest_region_hash","Price","Time
    order_deets = pd.concat([order_deets, order_dets], ignore_index=True)
print(order_dets)

for i in range(10,22):
    last=str(i)
    file_loc=file_loc[:- 2]+last
    # print(file_loc)

    order_dets = pd.read_csv(file_loc, sep='\t', header=None, names=["order_id","driver_id","passenger_id","start_region_hash","dest_region_hash","Price","Time
    order_deets = pd.concat([order_deets, order_dets], ignore_index=True)
✓ 2.0s                                                                                                                                    Python
```

In order_dets, 1 by 1 each file is being read while in order_deets each file gets appended. The for loop is used to modify the file location for each file. There are total 21 files, for the 1st 9 files

we have first for loop which replaces the last character in the file path while for the remaining 10 – 21 files, second for loop replaces the last 2 characters in the file path.

Then weather data is being read in the same way.

```
file_loc = "C:/Users/DELL/Downloads/ai_proj/training_data/training_data/weather_data/weather_data_2016-01-01"
size = len(file_loc)
weather_deets = pd.DataFrame(columns=["Time","Weather","Temprature","PM2.5"])
weather_dets= pd.DataFrame(columns=["Time","Weather","Temprature","PM2.5"])
for i in range(1,10):
    #file_loc = file_loc.replace(file_loc[size - 1], str(i))
    last=str(i)
    file_loc=file_loc[:- 1]+last
    print(file_loc)

    weather_dets= pd.read_csv(file_loc, sep='\t', header=None, names=["Time","Weather","Temprature","PM2.5"])
    weather_deets = pd.concat([weather_deets, weather_dets], ignore_index=True)

for i in range(10,22):
    last=str(i)
    file_loc=file_loc[:- 2]+last
    print(file_loc)

    weather_dets = pd.read_csv(file_loc, sep='\t', header=None, names=["Time","Weather","Temprature","PM2.5"])
    weather_deets = pd.concat([weather_deets, weather_dets], ignore_index=True)

print(weather_dets)
weather_deets.to_csv("all_weather.csv")
```

✓ 0.5s

In weather_dets, 1 by 1 each file is being read while in weather_deets each file gets appended. The for loop is used to modify the file location for each file.

```
file_loc = "C:/Users/DELL/Downloads/ai_proj/training_data/training_data/poi_data/poi_data"
size = len(file_loc)
#deets = pd.DataFrame(columns=["region_hash","poi_class"])


detss = pd.read_csv(file_loc, header=None)
# names=["region_hash","poi_class"])
detss=detss[0].str.split('\t',expand=True)
print(detss)
```

✓ 0.1s

Each column is stored in 1 index of detss. In detss[0], the region is stored. And in detss[1], the first value and feature following region is stored for each row.

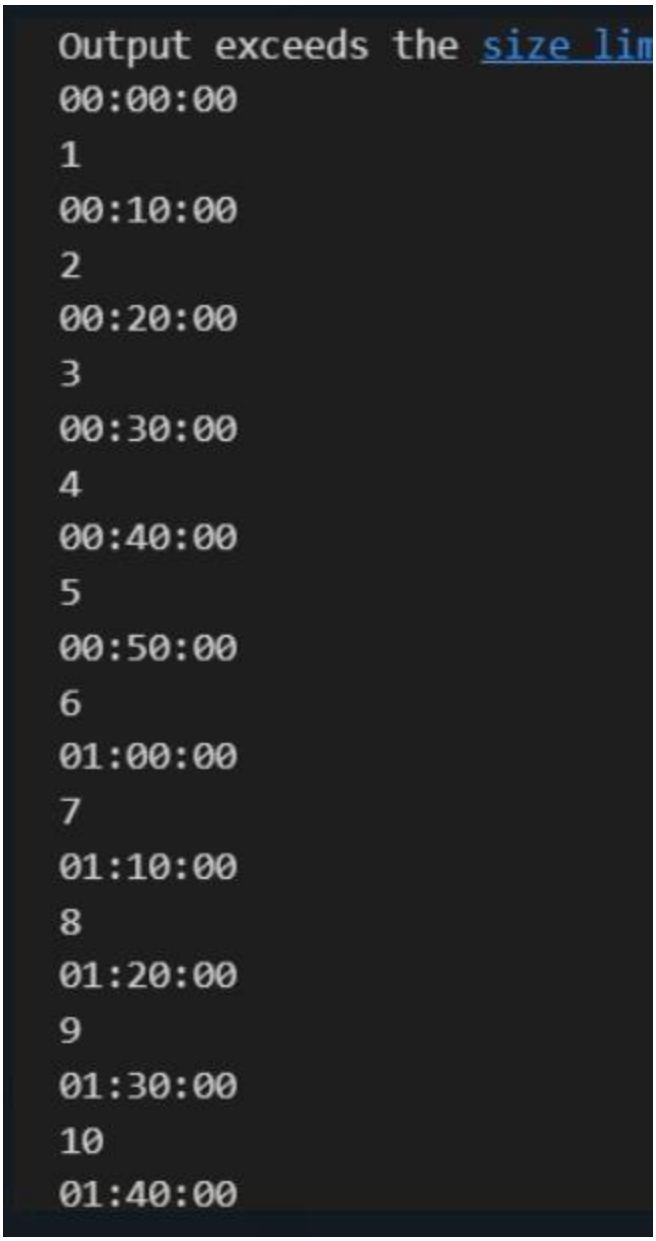
Step 2:

In next step, we are making time slots, which help us in assigning slot number(1 – 144). Slots are stored in var time_slots.

```python
import datetime
#iss main now we can go through file
# Set the start and end times
start_time = datetime.datetime(2023, 4, 19, 0, 0, 0)
end_time = datetime.datetime(2023, 4, 20, 0, 0, 0)

# Create a list to hold the time slots
time_slots = []

# Loop through 10 minute intervals
while start_time < end_time:
    time_slots.append(start_time.strftime("%H:%M:%S"))
    start_time += datetime.timedelta(minutes=10)

# # Print the time slots
i= 0
for slot in time_slots:
    print(slot)
    print(i)
    i=i+1
```

✓ 0.1s

Here is the screenshot of part of output:



Step 3:

Now we are adding columns named slots and date, and initializing both by 0. From the existing dataframe we take time column and store the date in the date and by using time and time_slots array along with catering and exception of last slot we assign slot numbers to each order based on the slot it was placed in.

```
#abb har uss ka ham har ek file kee nayee file banatay hain
#weather region start_Region_ka_count    uss_main_se_null_ka_count  konsa_time_slot
import time
import datetime
order_deets['Time']=pd.to_datetime(order_deets['Time'])
order_deets['slot'] = 0
order_deets['date'] = 0
for index, row in order_deets.iterrows():
    #abb humnay conditional insert karnaa hai new column
    current_time = row['Time'].time()
    current_date = row['Time'].date()
    current_date = current_date.strftime("%Y-%m-%d")
    found=False
    for the_slot in range(0, len(time_slots)-1):
        time_tuple1 = datetime.datetime.strptime(time_slots[the_slot], '%H:%M:%S')
        time_only1 = time_tuple1.time()
        time_tuple2 = datetime.datetime.strptime(time_slots[the_slot+1], '%H:%M:%S')
        time_only2 = time_tuple2.time()
        if(current_time.minute>=50 and current_time.hour==23 ):
            found=True
            order_deets.at[index, 'slot'] = 144
            order_deets.at[index, 'date'] = current_date

        if(current_time>=time_only1 and current_time<time_only2):
            # idhar we will have a while loop which will go throught this
            for i in range (0,10):
                if time_tuple1.minute==current_time.minute and time_tuple1.hour==current_time.hour:

                    order_deets.at[index, 'slot'] = the_slot+1
                    order_deets.at[index, 'date'] = current_date
                    found=True
                time_tuple1 += datetime.timedelta(minutes=1)
        if(found):
            break
print(order_deets)
```

Same steps are followed for weather_deets, here is the snippet of code:

```
import datetime
rows=0
weather_deets['Time']=pd.to_datetime(order_deets['Time'])
weather_deets['slot'] = 0
weather_deets['date'] = 0
for index, row in weather_deets.iterrows():
    rows=rows+1
    #abb humnay conditional insert karnaa hai new column
    current_time = row['Time'].time()
    current_date = row['Time'].date()
    current_date = current_date.strftime("%Y-%m-%d")
    found=False
    for the_slot in range(0, len(time_slots)-1):
        time_tuple1 = datetime.datetime.strptime(time_slots[the_slot], '%H:%M:%S')
        time_only1 = time_tuple1.time()
        time_tuple2 = datetime.datetime.strptime(time_slots[the_slot+1], '%H:%M:%S')
        time_only2 = time_tuple2.time()
        if(current_time.minute>=50 and current_time.hour==23 ):
            found=True
            weather_deets.at[index, 'slot'] = 144
            weather_deets.at[index, 'date'] = current_date
        elif(current_time>=time_only1 and current_time<time_only2):
            for i in range (0,10):
                if time_tuple1.minute==current_time.minute and time_tuple1.hour==current_time.hour:
                    # print("slot found")
                    weather_deets.at[index, 'slot'] = the_slot+1
                    weather_deets.at[index, 'date'] = current_date
                    #print(time_slots[i])
                    found=True
                time_tuple1 += datetime.timedelta(minutes=1)
        if(found):
            break
print("the rows are")
print(rows)
print(weather_deets)
weather_deets.to_csv('all_weather_with_date8.csv', index=False)
```

Step 4:

```
order_count = order_deets.groupby(["slot", "start_region_hash","date"])["order_id","driver_id"].count().reset_index()
```
✓ 0.4s

Now we are creating a data frame using a groupby function. We are getting the count of orders placed in a specific region on a specific date during a specific slot and how many were received by drivers and how many requests got entertained by them.

Step 5:

```
order_weather = pd.merge(order_count, weather_deets,on=['slot','date'], how='left')
```
✓ 0.1s

This merges the data of weather files and order file based on slot and date.

Step 6:

```
#now finding total facilities in that sector
order_weather['total_facilities_in_area']=0
for index, row in order_weather.iterrows():
    start_region_hash_value = row['start_region_hash']
    total=0
    for loc in range (0, len(detss[0])):
        if start_region_hash_value.strip()== detss[0][loc].strip():
            j=loc
            for i in range(1,152):
                if(detss[i][j]!=None):
                    feature_and_value=detss[i][j]
                    if feature_and_value != None:
                        feat,value=feature_and_value.split(':')
                        total=total+int(value)
                        order_weather.at[index, 'total_facilities_in_area'] = total
```

✓ 17m 39.8s

Step 7:

We are calculating the gap in this step, which is the difference of order_id and driver_id.

```
order_weather['gap'] = order_weather['order_id'] - order_weather['driver_id']
```

✓ 0.0s

Step 8:

In this step, we are renaming column names order_id as number_of orders and driver_id as requests_entertained.

```
order_weather = order_weather.rename(columns={'order_id': 'number_of_orders', 'driver_id': 'requests_entertained'})
```

✓ 0.0s

Step 9:

Now, we are merging the order_weather with the cluster_map to get start_region_hash based on region_hash.

```
order_weather = pd.merge(order_weather, cluster_map, left_on='start_region_hash', right_on='region_hash', how='left')
```

✓ 0.1s

Step 10:

Now we are training the model using linear regression. Here are the code and output snippet, we got:

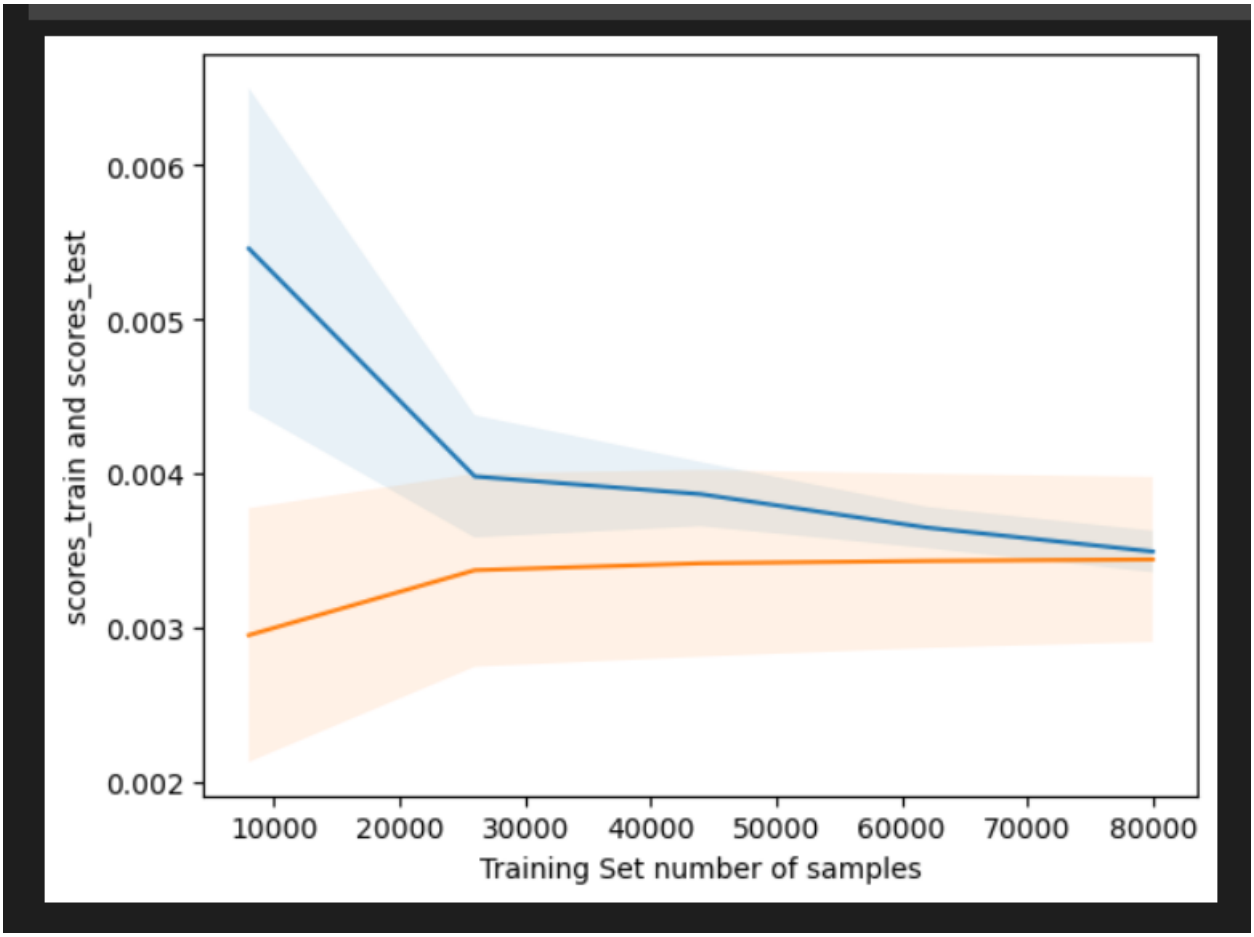
Code:

```
##################LINEAR REGRESSION#########################
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

data_f = pd.read_csv('wonderland12.csv')

X_trainn, X_test, y_trainn, y_test = train_test_split(
    data_f[['id_region', 'slot']], data_f['gap'], test_size=0.2, random_state=42)

encoder = OneHotEncoder(sparse=False)
X_train_encoded = encoder.fit_transform(X_trainn[['id_region']])
X_test_encoded = encoder.transform(X_test[['id_region']])

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_trainn[['slot']])
X_test_scaled = scaler.transform(X_test[['slot']])
trainXfin = np.concatenate([X_train_encoded, X_train_scaled], axis=1)
testXFin = np.concatenate([X_test_encoded, X_test_scaled], axis=1)

model = LinearRegression()
model.fit(trainXfin, y_trainn)

# Predict on the test data
pred_yi = model.predict(testXFin)
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(y_test, pred_yi)
print('mae:',mae)
```

✓ 3.0s

Output:

```
c:\Users\DELL\anaconda3\lib\s
  warnings.warn(
mae: 0.4660162940791283
```



Step 11:

As we didn't get the satisfactory output using linear regression model, so now we are trying to use a different model. So now we are training the model using Ridge. Here are the code and output snippets, we got:

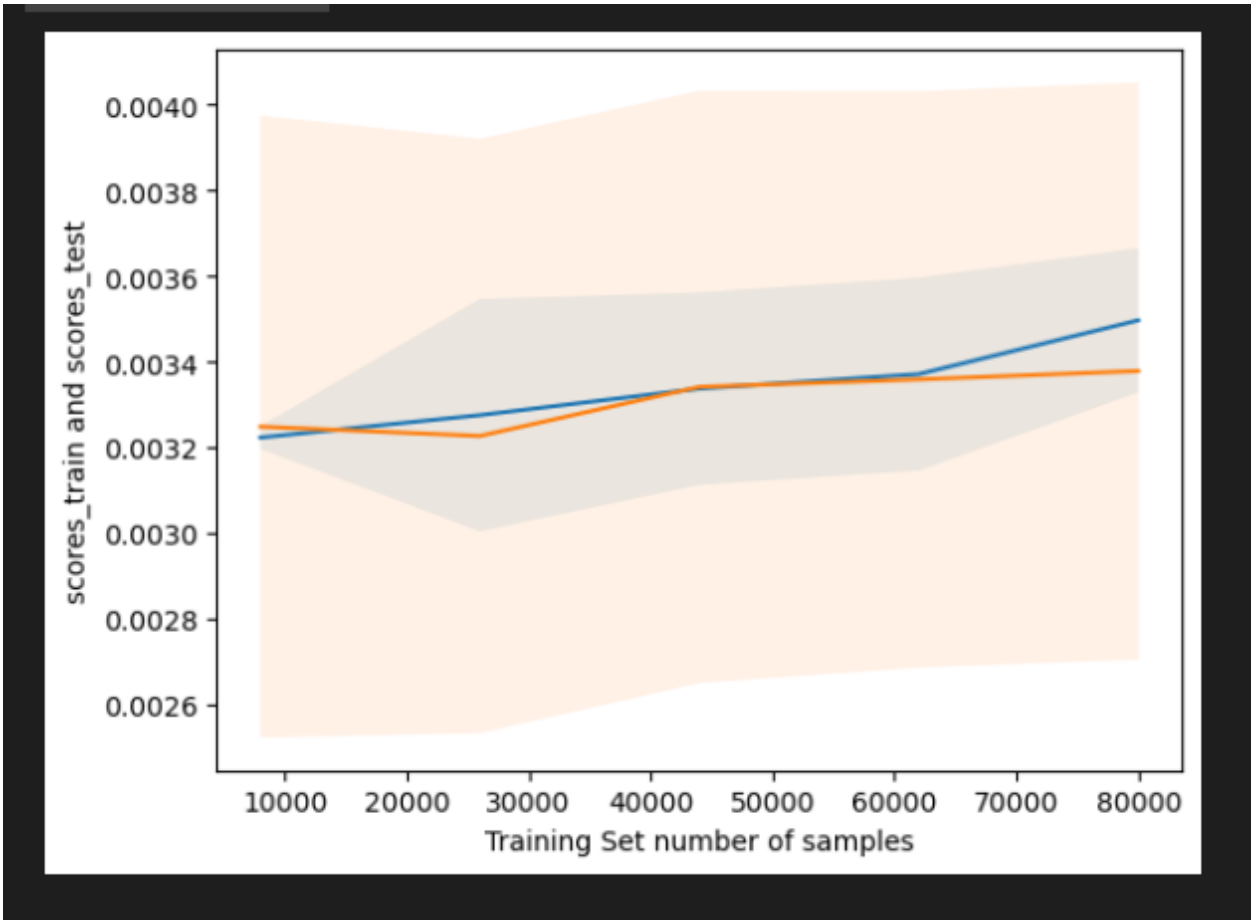
Code:

```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

data_f = pd.read_csv('wonderland12.csv')


X_trainn, X_test, y_trainn, y_test = train_test_split(
    data_f[['id_region', 'slot']], data_f['gap'], test_size=0.2, random_state=42)


encoder = OneHotEncoder(sparse=False)
X_train_encoded = encoder.fit_transform(X_trainn[['id_region']])
X_test_encoded = encoder.transform(X_test[['id_region']])


scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_trainn[['slot']])
X_test_scaled = scaler.transform(X_test[['slot']])
var = 1

trainXfin = np.concatenate([X_train_encoded, X_train_scaled], axis=var)
print("")
testXFin = np.concatenate([X_test_encoded, X_test_scaled], axis=1)

alpha = 1
model = Ridge(alpha=alpha)
model.fit(trainXfin, y_trainn)
pred_yi = model.predict(testXFin)
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(y_test, pred_yi)
print('mae:',mae)
```

✓ 0.7s

Output:

```
c:\Users\DELL\anaconda3\lib\s
  warnings.warn(

mae: 0.466025029600761
```



Step 12:

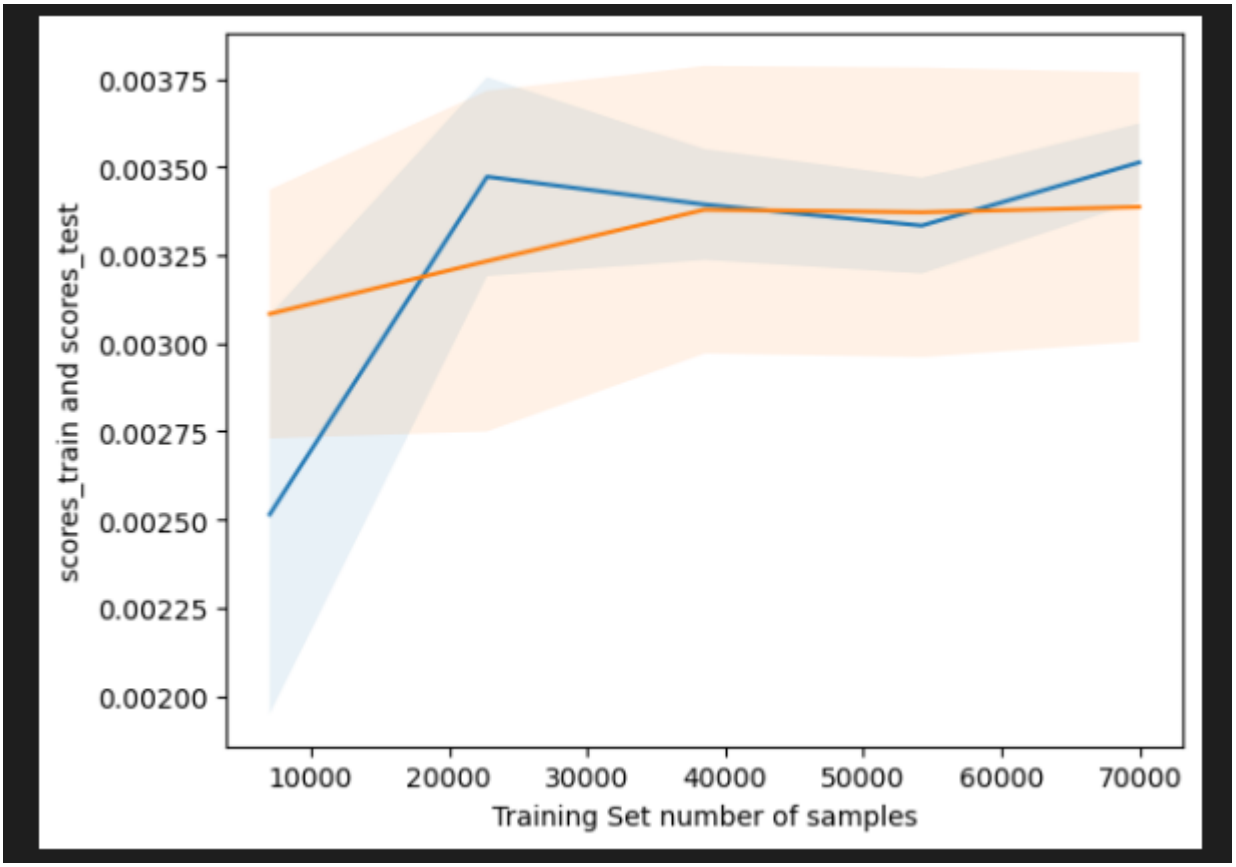
As we didn't get the satisfactory output using linear regression model and ridge, so now we are trying to use a different model. Now we are training the model using Decision Tree Classifier. Here are the code and output snippet, we got:

Code:

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder

checkDf = pd.read_csv('wonderland12.csv')
TempLabEnc = LabelEncoder()
data_f['id_region'] = TempLabEnc.fit_transform(data_f['id_region'])
X_trainn, X_test, y_trainn, y_test = train_test_split(data_f[['id_region', 'slot']], data_f['gap'], test_size=0.2, random_state=42)
ModClasf = DecisionTreeClassifier()
ModClasf.fit(X_trainn, y_trainn)
pred_yi = ModClasf.predict(X_test)
out_accuracy = accuracy_score(y_test, pred_yi)
print("accuracy is: ",out_accuracy)
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(y_test, pred_yi)
print('mae:',mae)

✓ 0.7s
```

Output:

```
accuracy is:  0.8977395479095819
mae: 0.20720144028805762
```



So after using different models for the training of the data, we can conclude that we got the best result from the last one i.e. Decision tree classifier. Because the mean absolute error we got from this is the least comparing to other two that are linear regression and ridge.