بسم الله الرحمن الرحيم

Professor : Dr.Bagher Babaali

Student : Ahmad Asadi

Student Number : 610303080

Homework 5 : Machine Learning

Question 3:

Step 1: Install Libraries

```
!pip install matplotlib seaborn
```

Step 2: Import Libraries

```
import pandas as pd
import numpy as np

import math

from io import StringIO

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.neighbors import KernelDensity

import matplotlib.pyplot as plt
import seaborn as sns
```

Pandas : Working with data frames and loading datasets into these frames

Numpy : have predefined functions for working with 2D arrays

math : using some calculation for calculating distance

train_test_split : for splitting the dataset into train and validation

LinearRegression : the linear regression model for training data

mean_squared_error : for calculating the MSE

r2_score : for calculating R2

matplotlib : showing charts

seaborn : showing charts more beautiful

Step 3: Get Dataset Pandas Frame from Github

```
train_set_url = "https://raw.githubusercontent.com/AsadiAhmad/House-Price-
Prediction/refs/heads/main/Dataset/train.csv"
test_set_url = "https://raw.githubusercontent.com/AsadiAhmad/House-Price-
Prediction/refs/heads/main/Dataset/test.csv"

pd.set_option('display.max_rows', None)

train_set = pd.read_csv(train_set_url)
test_set = pd.read_csv(test_set_url)
```

Step 4: Divide Dataset into Train_set and Validation_set

```
train_set, validation_set = train_test_split(train_set, test_size=0.2,
random_state=42)
```

Step 5: Calculate Pearson Correlation

```
def calculate_pearson_correlation(feature_list, target_list):
    feature_mean = sum(feature_list) / len(feature_list)
    target_mean = sum(target_list) / len(target_list)

    feature_diff = []
    for item in feature_list:
        feature_diff.append(item - feature_mean)

    target_diff = []
    for item in target_list:
        target_diff.append(item - target_mean)

    multiply_diff = []
    for index in range(len(feature_diff)):
        multiply_diff.append(feature_diff[index] * target_diff[index])

    square_feature_diff = []
    for item in feature_diff:
        square_feature_diff.append(item*item)

    square_target_diff = []
    for item in target_diff:
        square_target_diff.append(item*item)

    sum_multiply_diff = sum(multiply_diff)
    sum_square_feature_diff = sum(square_feature_diff)
    sum_square_target_diff = sum(square_target_diff)

    return sum_multiply_diff/math.sqrt(sum_square_feature_diff *
sum_square_target_diff)
```

```
all_cols = [train_set[col].tolist() for col in train_set.columns]
pearson_correlation = []
for anyList in all_cols[0:-1]:
    pearson_correlation.append(calculate_pearson_correlation(anyList,
all_cols[-1]))
```

```
pearson_correlation
```

```
[0.6754163075637131,
 0.1275110626317921,
 0.1354915336600122,
 -0.0402437579970883,
 -0.027798655756735374,
 -0.03254710712812415,
 -0.12915768317543397,
 -0.04409562601510458]
```

Step 6: Calculate Mutual Information with KDE

```
def mutual_information_kde(feature_list, target_list, bandwidth=0.1):
    feature_list = feature_list.reshape(-1, 1)
    target_list = target_list.reshape(-1, 1)
    all = np.hstack((feature_list, target_list))

    kde_joint = KernelDensity(bandwidth=bandwidth).fit(all)
    log_prob_joint = kde_joint.score_samples(all)

    kde_features = KernelDensity(bandwidth=bandwidth).fit(feature_list)
    log_prob_features = kde_features.score_samples(feature_list)

    kde_tagrget = KernelDensity(bandwidth=bandwidth).fit(target_list)
    log_prob_tagrget = kde_tagrget.score_samples(target_list)

    mi = np.mean(log_prob_joint - log_prob_features - log_prob_tagrget)
    return mi
```

```
mutual_information = []
for anyList in all_cols[0:-1]:
    mutual_information.append(mutual_information_kde(np.array(anyList),
np.array(all_cols[-1])))
```

```
mutual_information
```

```
[0.3825063785343686,
 0.07163783891708123,
 0.13469740882072623,
 0.028059220216915766,
 1.2674582645373336,
 0.10463400815830431,
 0.2302446607972519,
 0.310168458978906 44]
```

Step 7: Feature Selection

```
cols = list(train_set.columns)[0:-1]
cols_copy = cols.copy()
cols_copy2 = cols.copy()
pearson_correlation_copy = []
mutual_information_copy = mutual_information.copy()

for element in pearson_correlation:
    if element < 0:
        pearson_correlation_copy.append(element * -1)
    else:
        pearson_correlation_copy.append(element)

pearson_correlation_features = []
mutual_information_features = []
for index in range(4):
    idx = pearson_correlation.index(max(pearson_correlation))
    pearson_correlation_features.append(cols_copy[idx])
    del cols_copy[idx]
    del pearson_correlation[idx]
    idx2 = mutual_information.index(max(mutual_information))
    mutual_information_features.append(cols_copy2[idx2])
    del cols_copy2[idx2]
    del mutual_information[idx2]
```

```
pearson_correlation_features
['MedInc', 'AveRooms', 'HouseAge', 'Population']
```

```
mutual_information_features
['Population', 'MedInc', 'Longitude', 'Latitude']
```

Step 8: Train Linear regression model with Pearson Correlation Features

```
features_train_pc = train_set[pearson_correlation_features]
target_train_pc = train_set['MedHouseVal']

features_validation_pc = validation_set[pearson_correlation_features]
target_validation_pc = validation_set['MedHouseVal']
```

```
model_pc = LinearRegression()
model_pc.fit(features_train_pc, target_train_pc)
```

```
target_validation_predict_pc = model_pc.predict(features_validation_pc)
```

Step 9: Train Linear regression model with Mutual Information Features

```
features_train_mi = train_set[mutual_information_features]
target_train_mi = train_set['MedHouseVal']

features_validation_mi = validation_set[mutual_information_features]
target_validation_mi = validation_set['MedHouseVal']
```

```
model_mi = LinearRegression()
model_mi.fit(features_train_mi, target_train_mi)
```

```
target_validation_predict_mi = model_mi.predict(features_validation_mi)
```

Step 10: Calculate measures for PC and MI

```
mse_pc = mean_squared_error(target_validation_pc,
target_validation_predict_pc)
r2_pc = r2_score(target_validation_pc, target_validation_predict_pc)
```

```
mse_mi = mean_squared_error(target_validation_mi,
target_validation_predict_mi)
r2_mi = r2_score(target_validation_mi, target_validation_predict_mi)
```

```
mse_pc
```

```
0.6977652123033251
```

```
r2_pc
```

```
0.4794349527562398
```

```
mse_mi
```

```
0.6043053087589882
```

```
r2_mi
```

```
0.5491603535731632
```

Step 11: Mutual Information Features works better!

```
features_test_mi = test_set[mutual_information_features]
target_test_predict_mi = model_mi.predict(features_test_mi)
test_set['MedHouseValPredicted'] = target_test_predict_mi
```

Step 12: Export the predicted EXCEL

```
test_set.to_csv('Predications2.csv', index=False)
```

Here is more info about the project in the Github repo!

https://github.com/AsadiAhmad/House-Price-Prediction