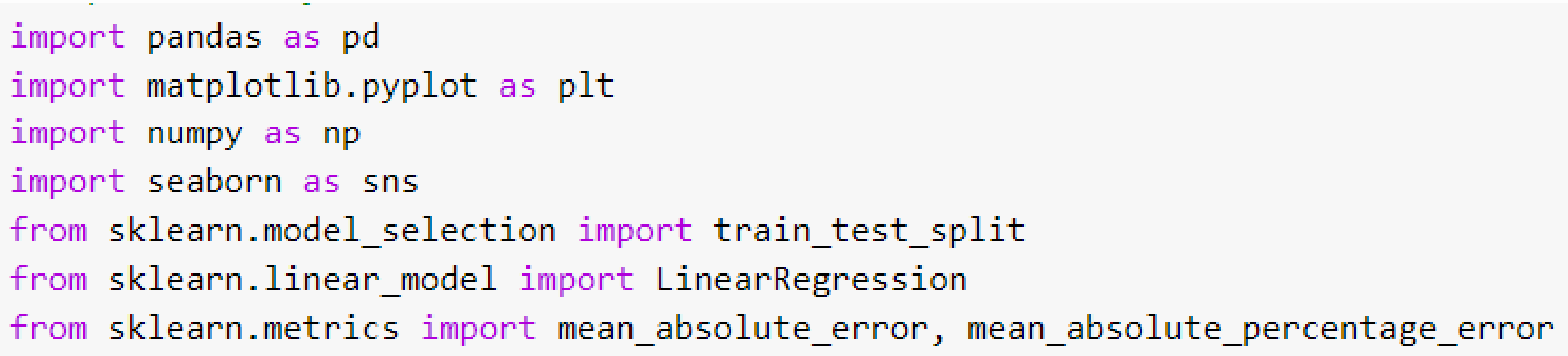
Prediction Modeling with Linear Regression

Anita Mila Oktafani

Data Preparation with Python

- Import Library
- Load Data
- Data Understanding
- Data Cleansing

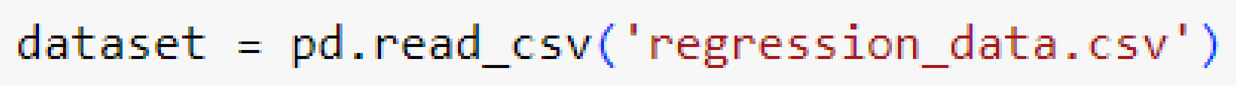
Import Library



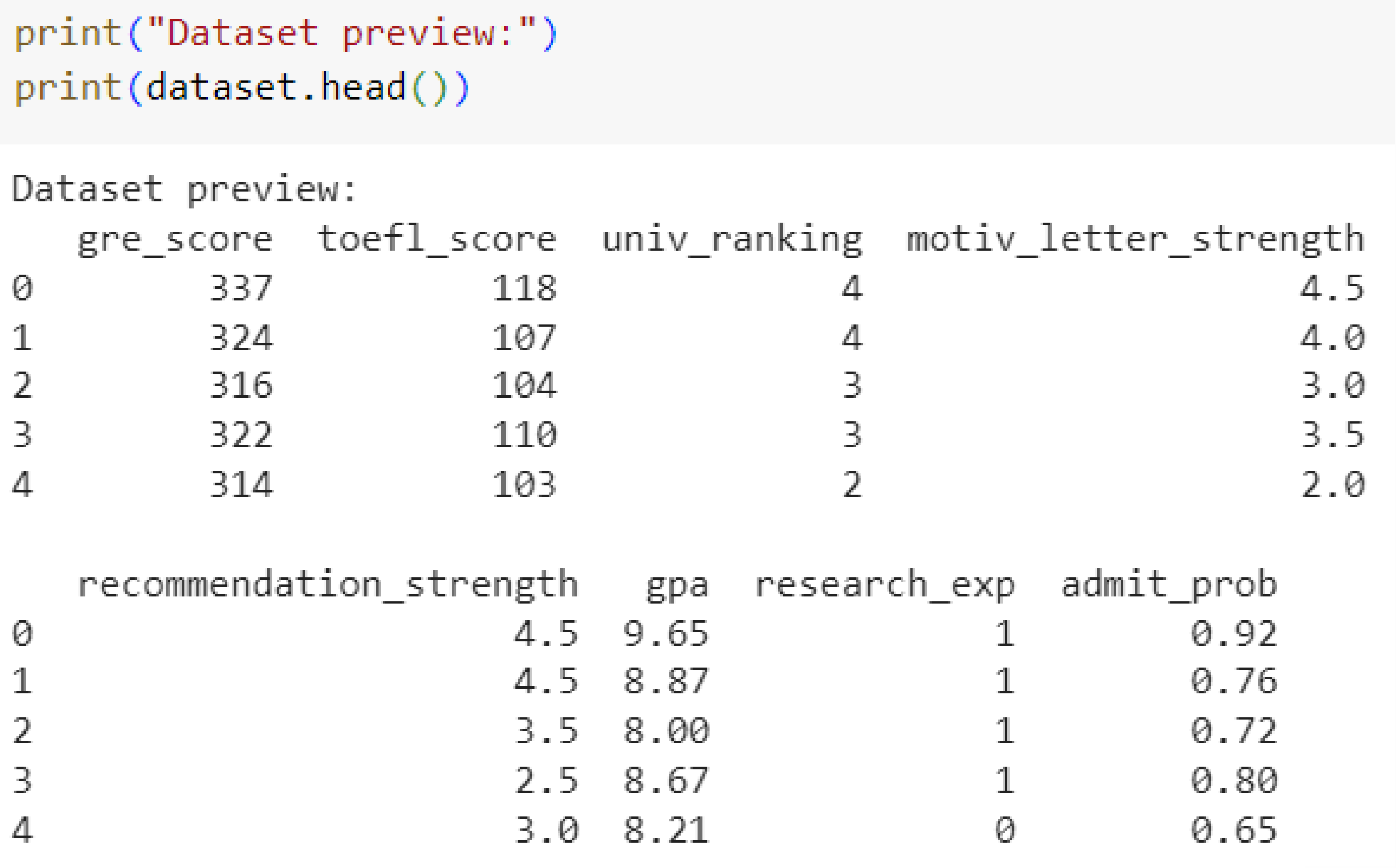
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error
```

Load Dataset

Dataset: Admit Probability



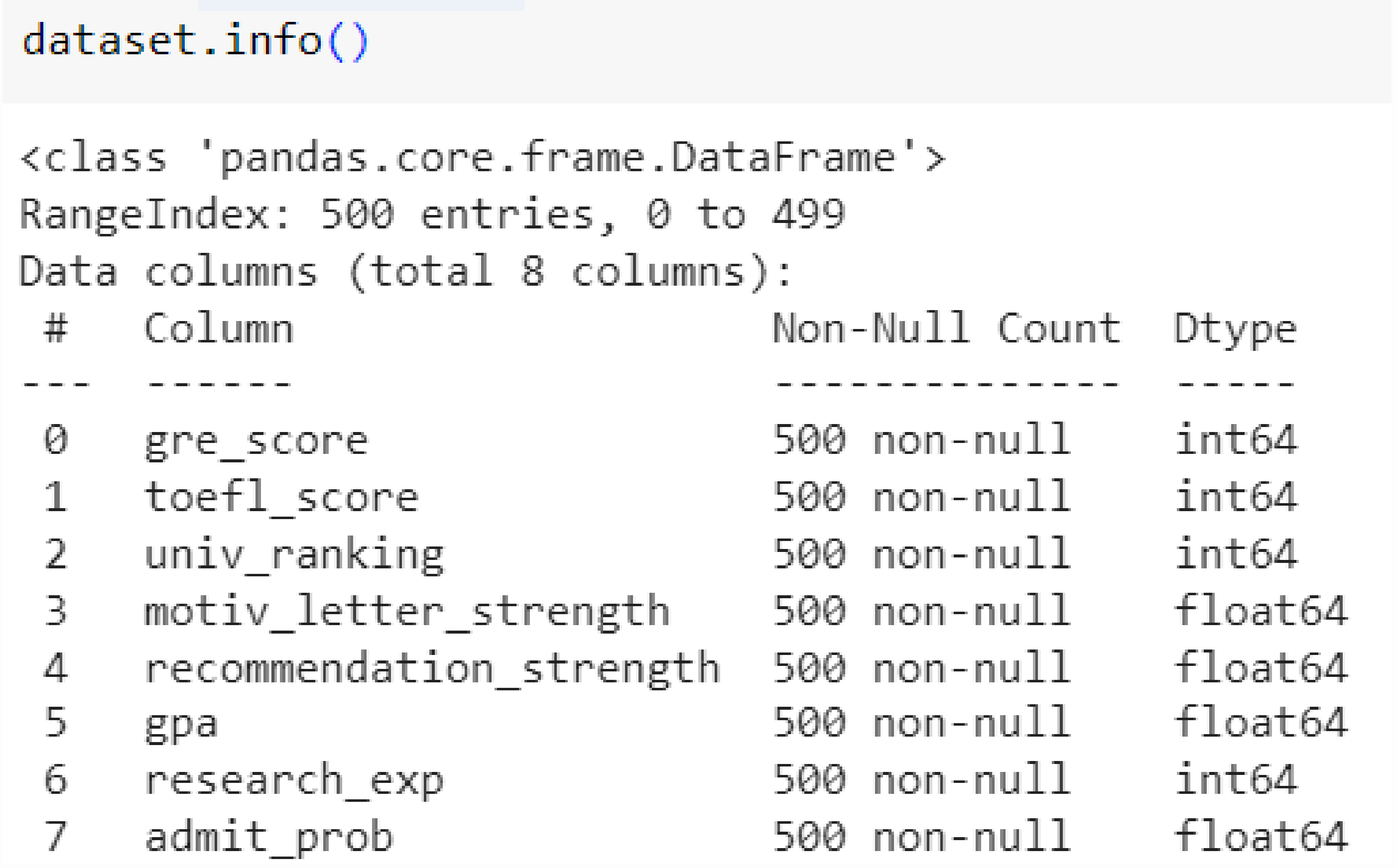
```
dataset = pd.read_csv('regression_data.csv')
```



```
print("Dataset preview:")
print(dataset.head())
```

```
Dataset preview:
   gre_score  toefl_score  univ_ranking  motiv_letter_strength  \
0        337          118             4                    4.5   
1        324          107             4                    4.0   
2        316          104             3                    3.0   
3        322          110             3                    3.5   
4        314          103             2                    2.0   

   recommendation_strength   gpa  research_exp  admit_prob  
0                      4.5  9.65             1        0.92  
1                      4.5  8.87             1        0.76  
2                      3.5  8.00             1        0.72  
3                      2.5  8.67             1        0.80  
4                      3.0  8.21             0        0.65  
```

Data Understanding



```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   gre_score                500 non-null    int64  
 1   toefl_score              500 non-null    int64  
 2   univ_ranking             500 non-null    int64  
 3   motiv_letter_strength    500 non-null    float64
 4   recommendation_strength  500 non-null    float64
 5   gpa                      500 non-null    float64
 6   research_exp             500 non-null    int64  
 7   admit_prob               500 non-null    float64
```

gre_score Score of GRE (Graduate Records Examination)

toefl_score Score of TOEFL (Test of English as a Foreign Language)

univ_ranking University ranking

motiv_letter_strength Scale of how strong the motivation letter is

recommdendation_strength Scale of how strong the recommendation is

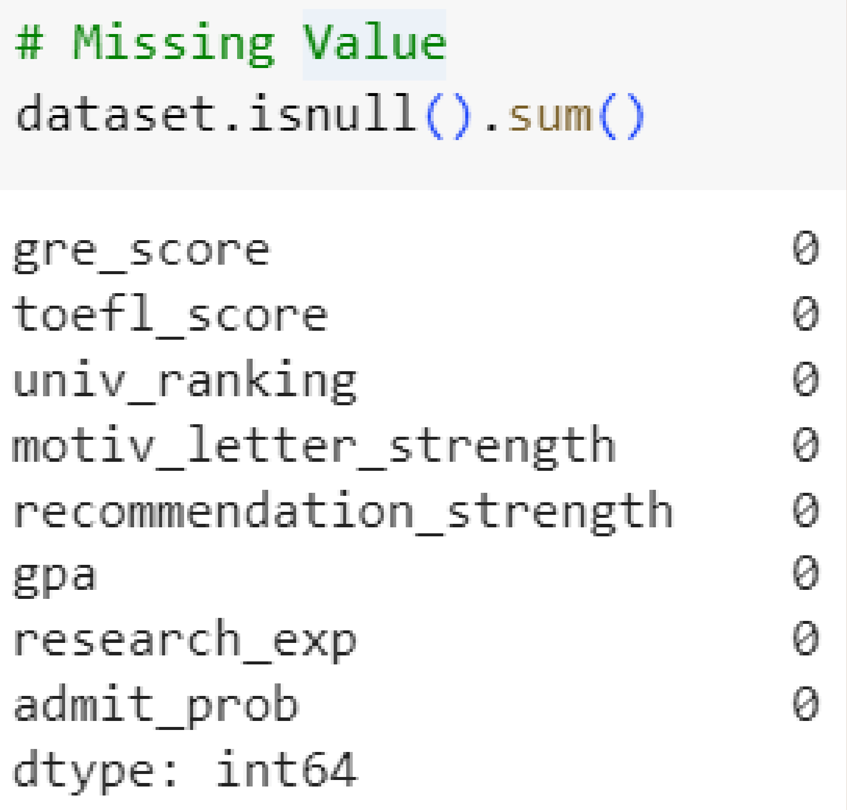
gpa Final grade of lecture

research_exp How much experience in conducting research

admit_prob The probability of these values being recognized

Data Cleansing

Missing Value

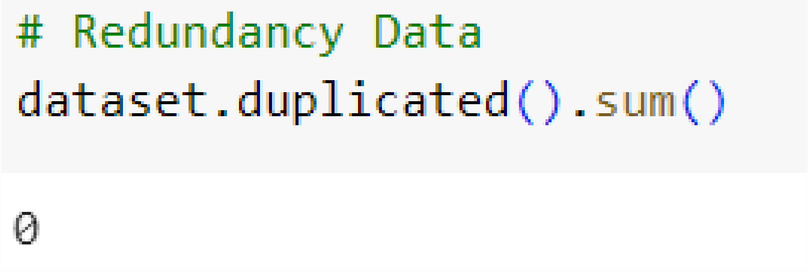


```
# Missing Value
dataset.isnull().sum()
```

```
gre_score                   0
toefl_score                 0
univ_ranking                0
motiv_letter_strength       0
recommendation_strength     0
gpa                         0
research_exp                0
admit_prob                  0
dtype: int64
```

There are no missing value detected

Duplicate Data

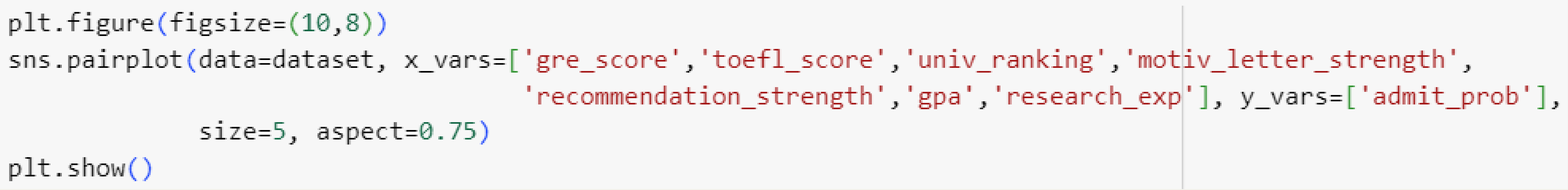


```
# Redundancy Data
dataset.duplicated().sum()
```

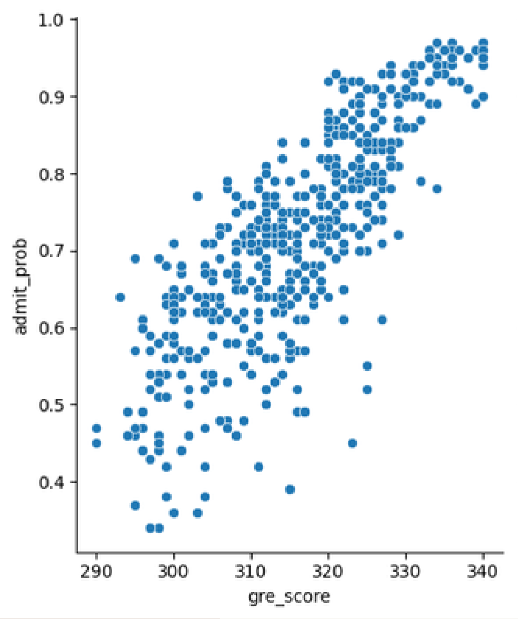
```
0
```

There are no duplicate data detected

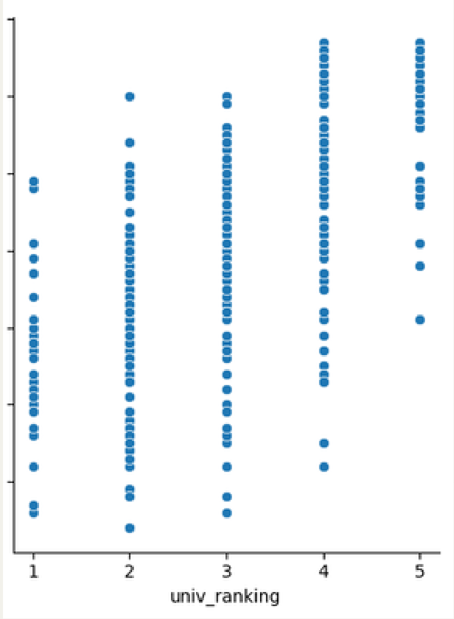
Data Correlation



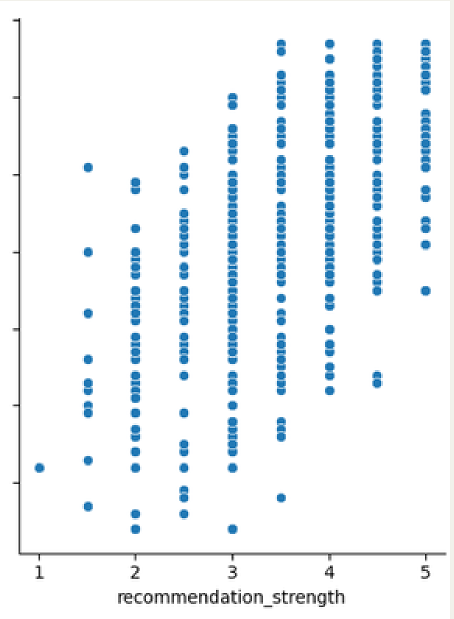
```
plt.figure(figsize=(10,8))
sns.pairplot(data=dataset, x_vars=['gre_score','toefl_score','univ_ranking','motiv_letter_strength',
                                   'recommendation_strength','gpa','research_exp'], y_vars=['admit_prob'],
             size=5, aspect=0.75)
plt.show()
```



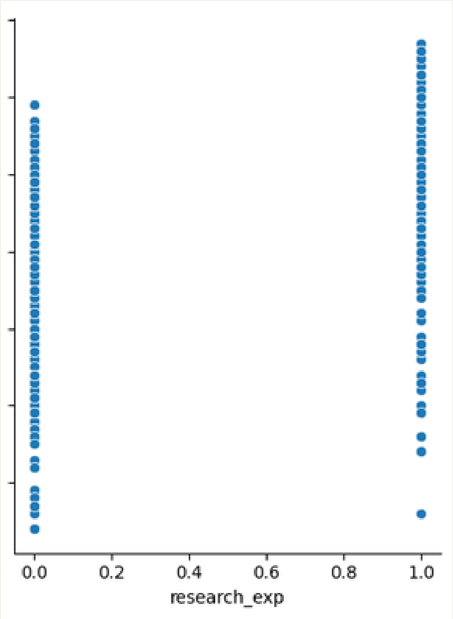
Strong positive correlation



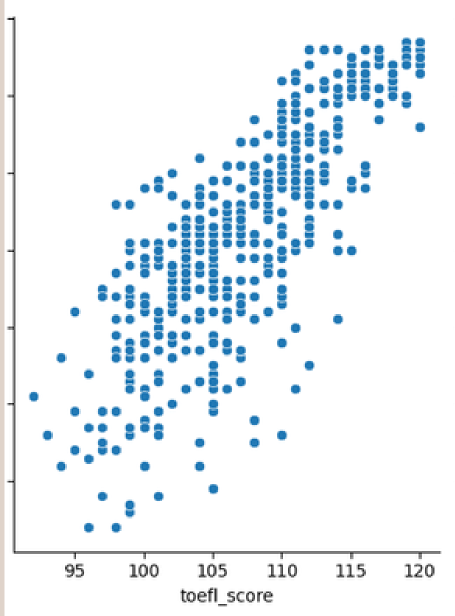
Weak positive correlation



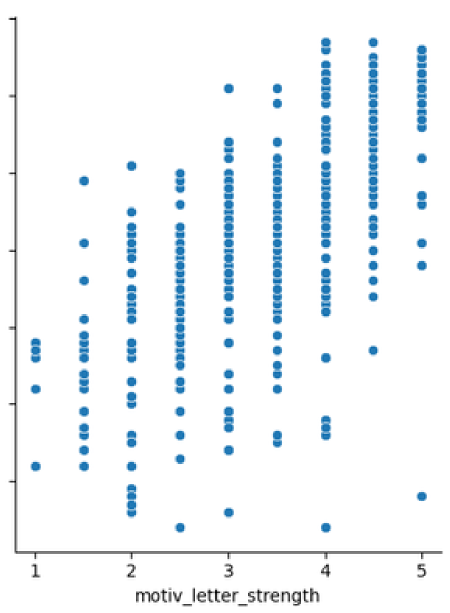
Weak positive correlation



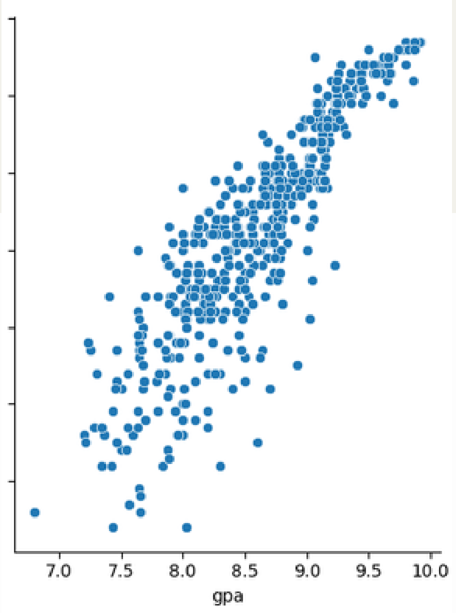
Weak positive correlation



Strong positive correlation

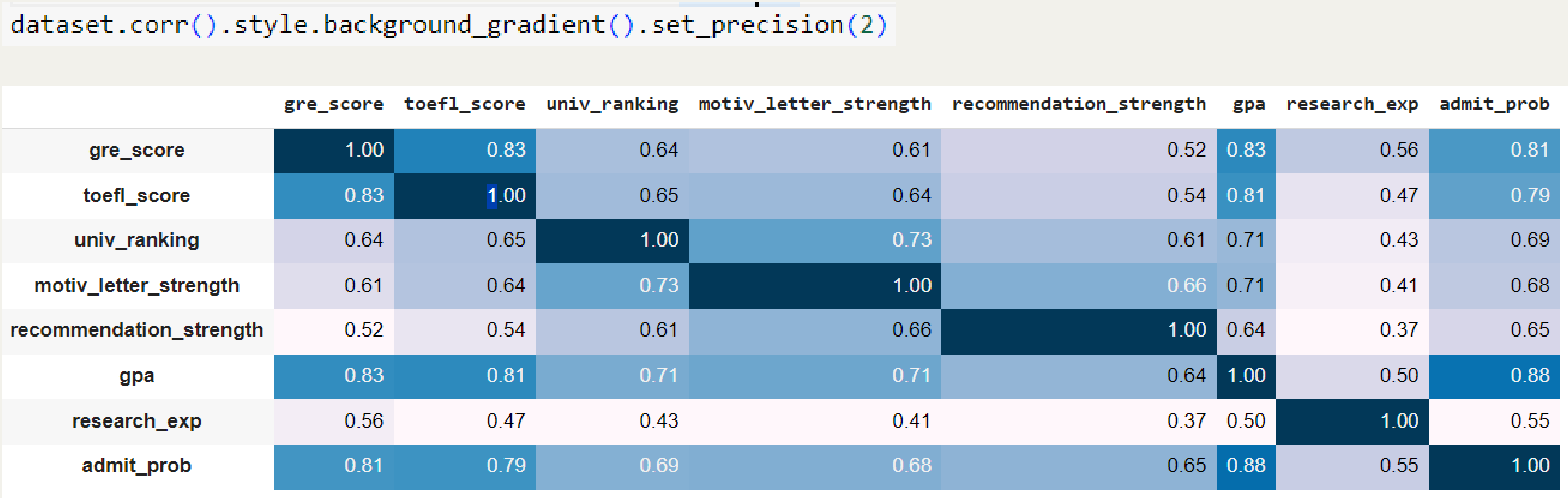


Weak positive correlation



Strong positive correlation

Data Correlation



```
dataset.corr().style.background_gradient().set_precision(2)
```

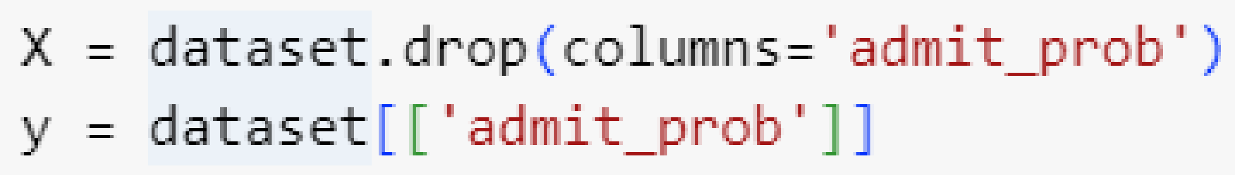
	gre_score	toefl_score	univ_ranking	motiv_letter_strength	recommendation_strength	gpa	research_exp	admit_prob
gre_score	1.00	0.83	0.64	0.61	0.52	0.83	0.56	0.81
toefl_score	0.83	1.00	0.65	0.64	0.54	0.81	0.47	0.79
univ_ranking	0.64	0.65	1.00	0.73	0.61	0.71	0.43	0.69
motiv_letter_strength	0.61	0.64	0.73	1.00	0.66	0.71	0.41	0.68
recommendation_strength	0.52	0.54	0.61	0.66	1.00	0.64	0.37	0.65
gpa	0.83	0.81	0.71	0.71	0.64	1.00	0.50	0.88
research_exp	0.56	0.47	0.43	0.41	0.37	0.50	1.00	0.55
admit_prob	0.81	0.79	0.69	0.68	0.65	0.88	0.55	1.00

- It can be seen that **gre_score, toefl_score** and **gpe** has a very **strong positive** linear relationship with admit_prob when compared to the others with a value almost close to 1

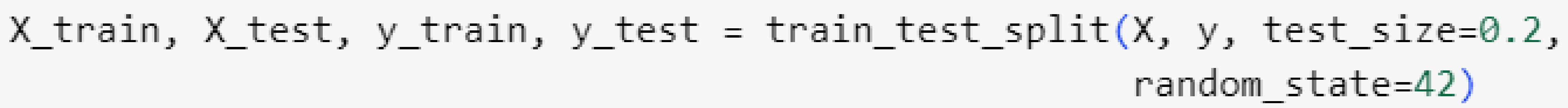
Regression Linear

- **Split Data**
- **Modelling**
- **Evaluation Model**
- **Visualization**

Split Data

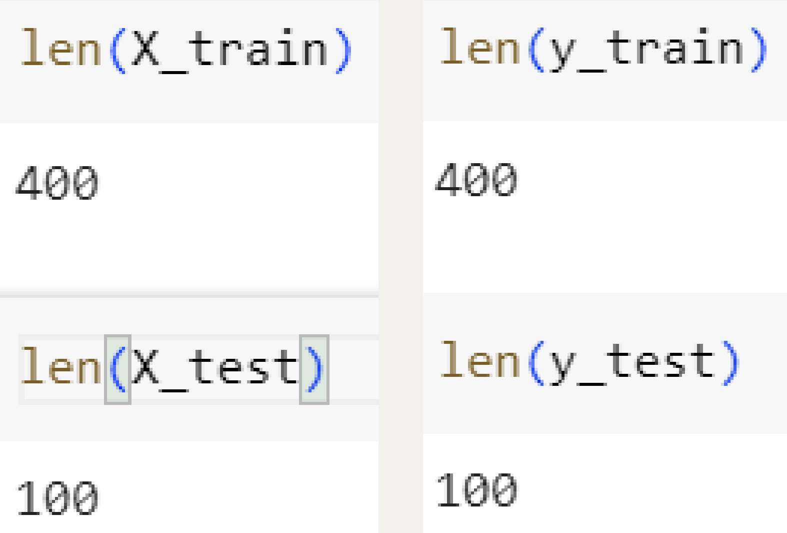


```
X = dataset.drop(columns='admit_prob')
y = dataset[['admit_prob']]
```



```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=42)
```

- The dataset is divided into 2, namely **training data** and **testing data**
- The dataset is divided using a proportion of 80 : 20, with 80% training data and 20% test data



```
len(X_train)

400
```

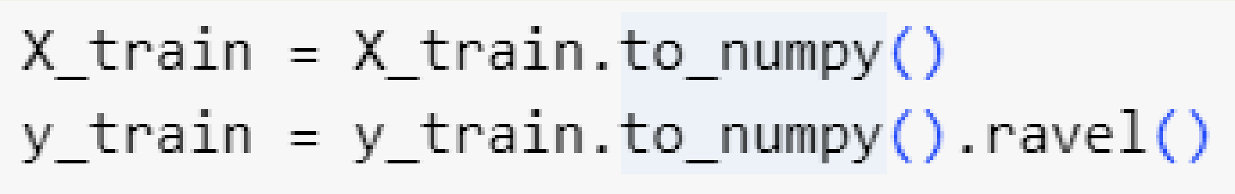
```
len(y_train)

400
```

80% of the training data is 400 data

```
len(X_test)

100
```

```
len(y_test)

100
```

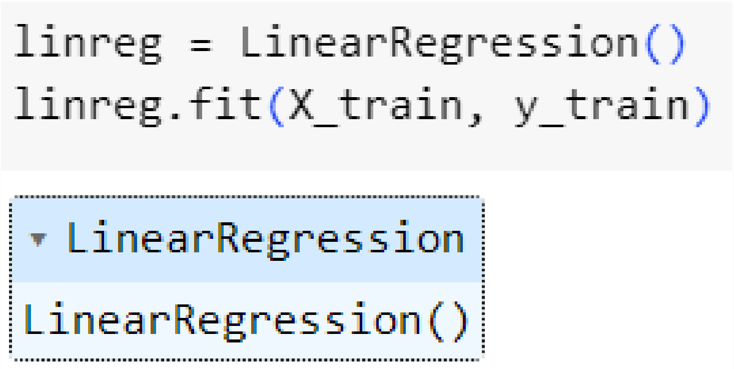
20% of the testing data is 100 data



```
X_train = X_train.to_numpy()
y_train = y_train.to_numpy().ravel()
```

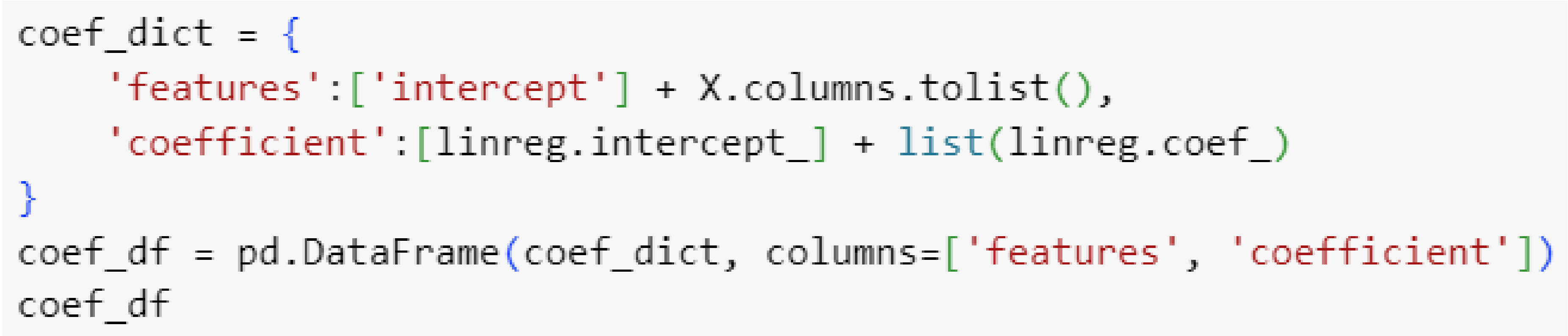
The dataset is still in dataframe format, therefore it needs to be converted to Numpy so it can be processed by Sklearn

Modeling

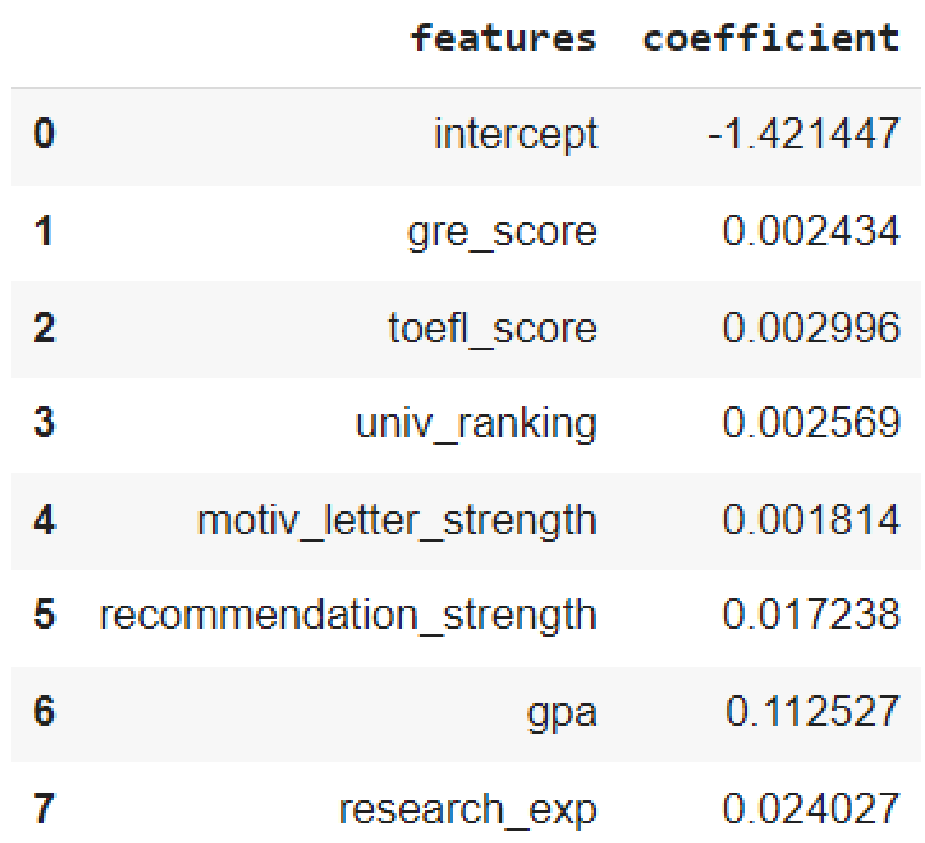


```
linreg = LinearRegression()
linreg.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```



```
coef_dict = {
    'features':['intercept'] + X.columns.tolist(),
    'coefficient':[linreg.intercept_] + list(linreg.coef_)
}
coef_df = pd.DataFrame(coef_dict, columns=['features', 'coefficient'])
coef_df
```

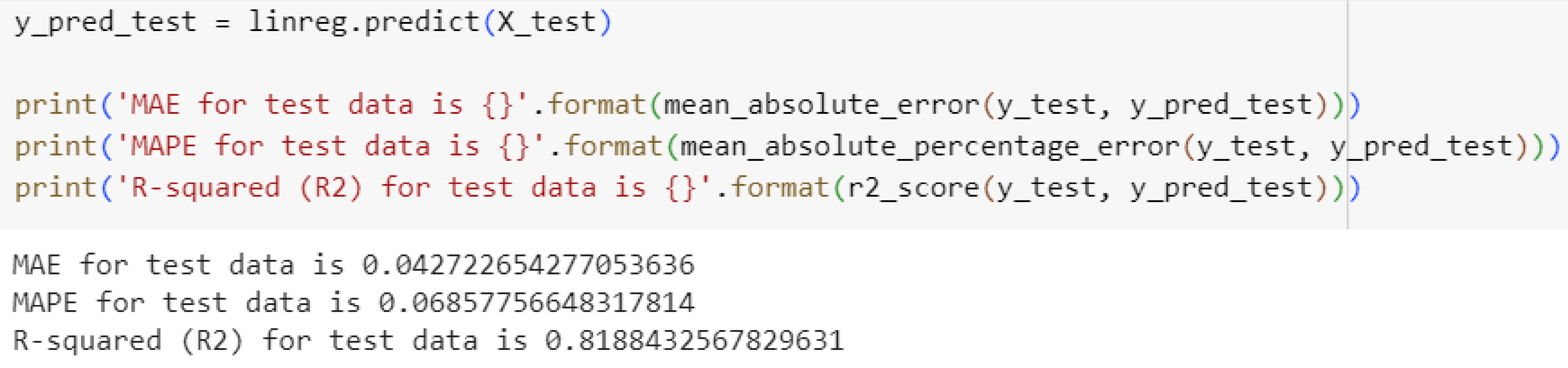


	features	coefficient
0	intercept	-1.421447
1	gre_score	0.002434
2	toefl_score	0.002996
3	univ_ranking	0.002569
4	motiv_letter_strength	0.001814
5	recommendation_strength	0.017238
6	gpa	0.112527
7	research_exp	0.024027

From the values below, the mathematical model can be written as follows

$$Y = -1.421447 + 0.002434\ x1 + 0.002996\ x2 + 0.002569\ x3 + 0.001814\ x4 + 0.017238\ x5 + 0.112527\ x6 + 0.024027\ x7$$

Create Prediction and Evaluation on Test Data



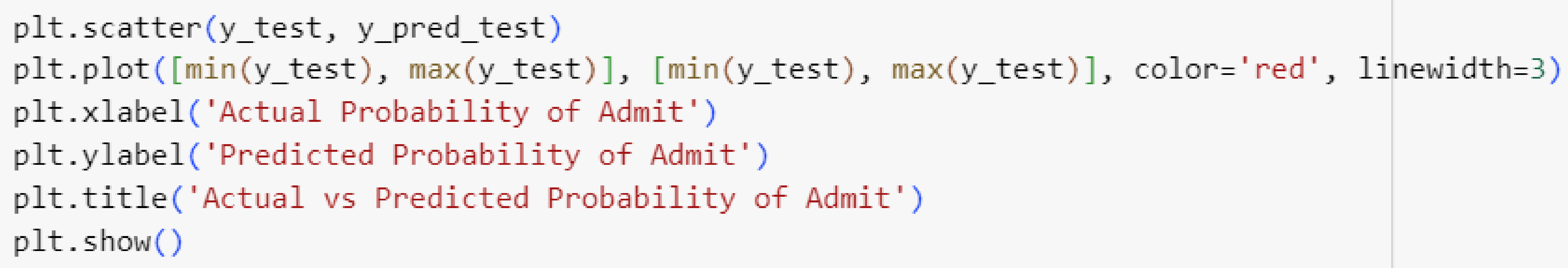
```
y_pred_test = linreg.predict(X_test)

print('MAE for test data is {}'.format(mean_absolute_error(y_test, y_pred_test)))
print('MAPE for test data is {}'.format(mean_absolute_percentage_error(y_test, y_pred_test)))
print('R-squared (R2) for test data is {}'.format(r2_score(y_test, y_pred_test)))
```

```
MAE for test data is 0.042722654277053636
MAPE for test data is 0.06857756648317814
R-squared (R2) for test data is 0.8188432567829631
```

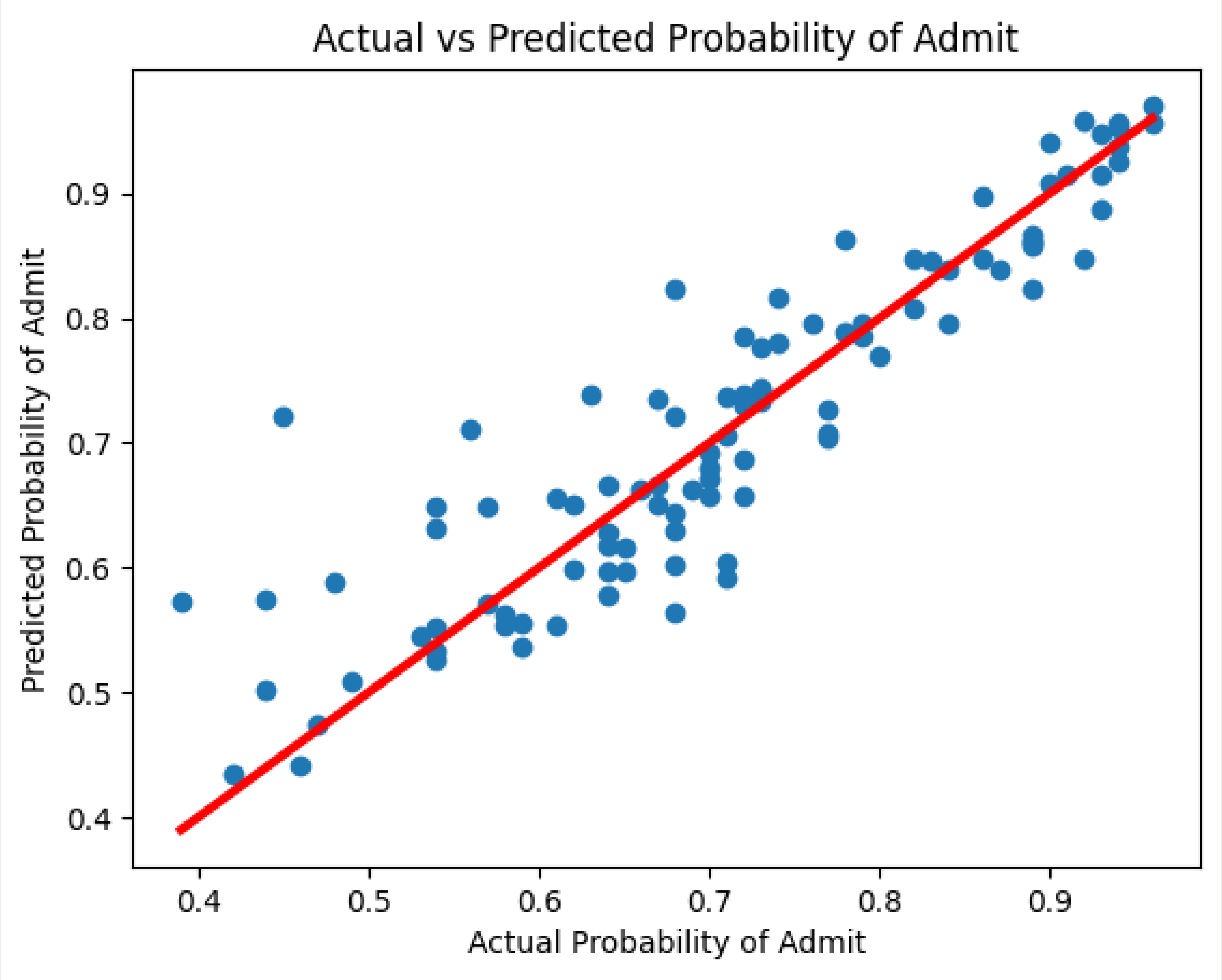
- The **MAE** value of 0.04272 means that the average model prediction error is around **0.04227** units of the target variable scale.
- The **MAPE** value of 0.06857 means that the average relative error of the model prediction is around 6.857%, which indicates a good level of accuracy.
- Both (MAE and MAPE) have low values, thus indicating that the model **has good performance** in predicting target values.

- The **R-squared** value of 0.8188 means that 80% of the model **can explain the data well**, also provides an indication that the model can **provide more accurate predictions**.

Visualization of the Result



```
plt.scatter(y_test, y_pred_test)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linewidth=3)
plt.xlabel('Actual Probability of Admit')
plt.ylabel('Predicted Probability of Admit')
plt.title('Actual vs Predicted Probability of Admit')
plt.show()
```



The points are distributed around the diagonal line (actual values), this shows that the model has **consistent performance** in predicting test data.

Thank You

Linkedin:
https://www.linkedin.com/in/anitamilaoktafani/

Github:
https://github.com/anitamila/Python_Regression-Linear