Name: Gore Aniket Machhindra

Email: aniket.m.gore.1901@gmail.com

Role: Machine Learning Intern

Task 4: Housing Prices Prediction Project

Content -

- Importing Libraries
- Importing the Dataset
- Initializing the DataFrame
- Adding Feature Names to the dataframe
- Adding target variable to dataframe
- Checking Missing Values
- Describe the Data
- Correlation
- Heatmap
- Splitting the Data
- Importing random Forest regressor
- Model Prediction on Training Data
- Model Evaluation
- Visualizing the differences between actual prices and predicted values
- Conclusion

Importing Libraries

```
import pandas as pd
import numpy as np
from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Importing the Dataset

```
from sklearn.datasets import load_boston
boston = load_boston()
```

Initializing the Dataframe

```
data = pd.DataFrame(boston.data)
```

```
data.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

Adding Feature Names to the dataframe

```
data.columns = boston.feature_names
data.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTA
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.9
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.1
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.0
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.9
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.3

Adding target variable to dataframe

```
data['PRICE'] = boston.target
```

```
data.shape
```

```
(506, 14)
```

```
data.columns
```

```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'PRICE'],
      dtype='object')
```

```
data.dtypes
```

```
CRIM       float64
ZN         float64
INDUS      float64
CHAS       float64
NOX        float64
RM         float64
AGE        float64
DIS        float64
RAD        float64
TAX        float64
PTRATIO    float64
B          float64
LSTAT      float64
PRICE      float64
dtype: object
```

Checking Missing Values

```
data.isnull().sum()
```

```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
```

```
PRICE     0
dtype: int64
```

Describe the Data

```
data.describe()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000

Correlation

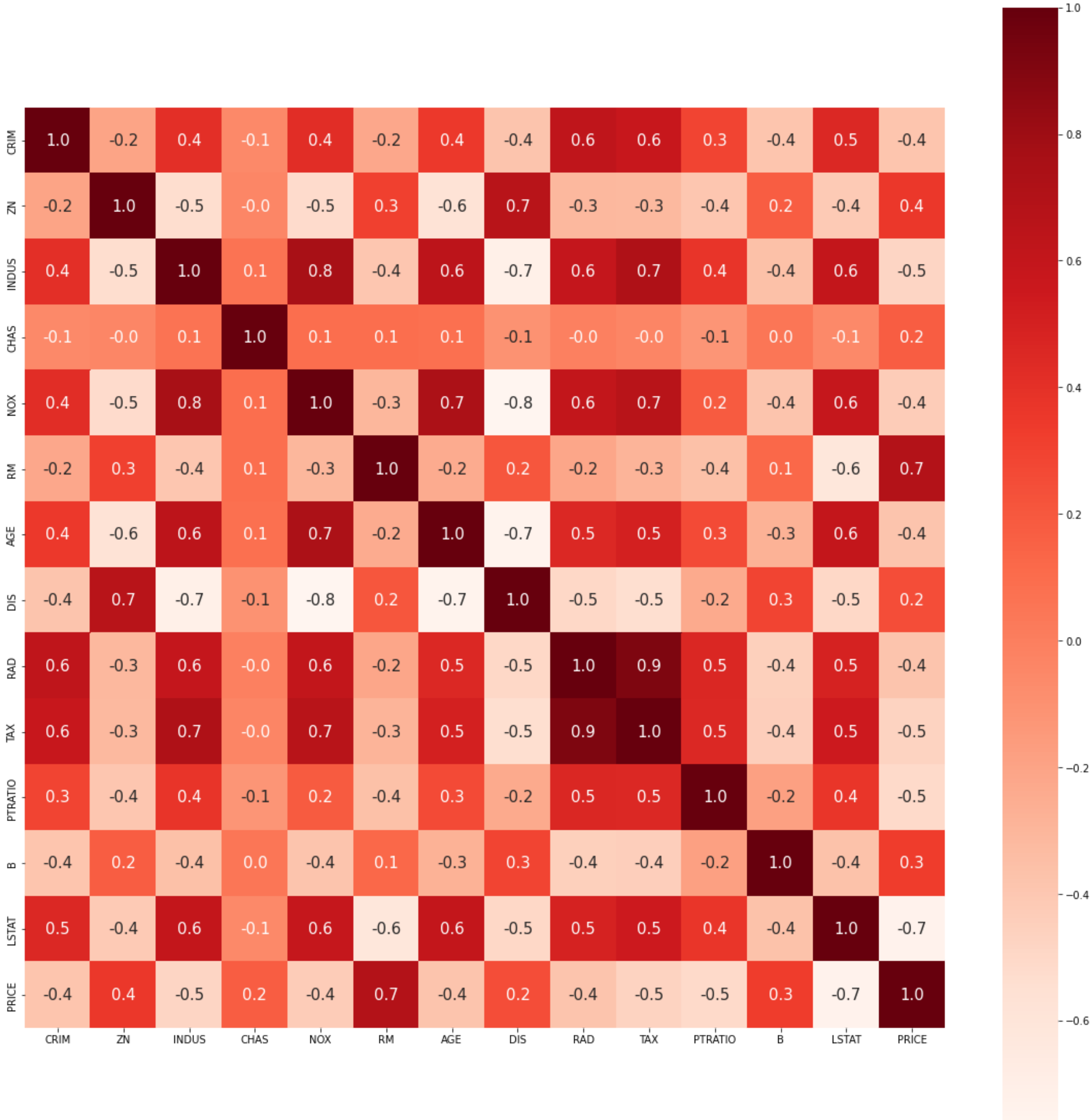
```
corr = data.corr()
corr.shape
```

```
(14, 14)
```

Heatmap

```
plt.figure(figsize=(20,20))
sns.heatmap(corr, cbar=True, square= True, fmt='.1f', annot=True, annot_kws={'size':15}, c
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f2fb7570b90>



Splitting the Data

```
X = data.drop(['PRICE'], axis = 1)
y = data['PRICE']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.3, random_state = 4
```

Importing random Forest regressor

```
from sklearn.ensemble import RandomForestRegressor
reg = RandomForestRegressor()
reg.fit(X_train, y_train)
```

```
RandomForestRegressor()
```

Model Prediction on Training Data

```
y_pred = reg.predict(X_train)
```

Model Evaluation

```
print('R^2:',metrics.r2_score(y_train, y_pred))
```

```
R^2: 0.9786402541948575
```

```
print('Adjusted R^2:',1 - (1-metrics.r2_score(y_train, y_pred))*(len(y_train)-1)/(len(y_tr
```

```
Adjusted R^2: 0.9778235580317196
```

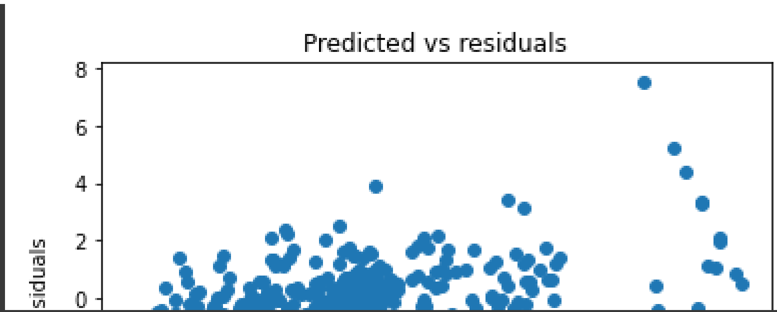
```
print('MAE:',metrics.mean_absolute_error(y_train, y_pred))
print('MSE:',metrics.mean_squared_error(y_train, y_pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_train, y_pred)))
```

```
MAE: 0.8485790960451979
MSE: 1.6077657881355942
RMSE: 1.2679770455870225
```

Visualizing the differences between actual prices and predicted values

```
plt.scatter(y_train, y_pred)
plt.xlabel("Prices")
plt.ylabel("Predicted prices")
plt.title("Prices vs Predicted prices")
plt.show()
```


```
plt.scatter(y_pred,y_train-y_pred)
plt.title("Predicted vs residuals")
plt.xlabel("Predicted")
plt.ylabel("Residuals")
plt.show()
```



Conclusion

- Random Forest regressor is works best for this dataset
- R2 Score is 97% Accurate on this Dataset
- Adjusted R2 Score is 97% Accurate on this Dataset

✓ 0s completed at 12:18 PM