REPORT

INTRODUCTION

House Prices Project is one of well known datasets. The target is to predict a house price with many explanatory variables, such as CRIM, Age, RM and so on. Basically, the dataset is made of train and test data sets. With 13 variables, I have to build my model to forecast individual house price and submit my prediction over test dataset. Let's start!

DATA DESCRIPTION

The Boston House prediction training dataset has 351 rows and 14 columns among which the last feature “medv” is the target feature. The following describes the dataset columns:

- CRIM - per capita crime rate by town
- ZN - proportion of residential land zoned for lots over 25,000 sq. ft.
- INDUS - proportion of non-retail business acres per town.
- CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
- NOX - nitric oxides concentration (parts per 10 million)
- RM - average number of rooms per dwelling
- AGE - proportion of owner-occupied units built prior to 1940
- DIS - weighted distances to five Boston employment centers
- RAD - index of accessibility to radial highways
- TAX - full-value property-tax rate per $10,000
- PTRATIO - pupil-teacher ratio by town
- B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
- LSTAT - % lower status of the population
- MEDV - Median value of owner-occupied homes in $1000's

MY APPROACH

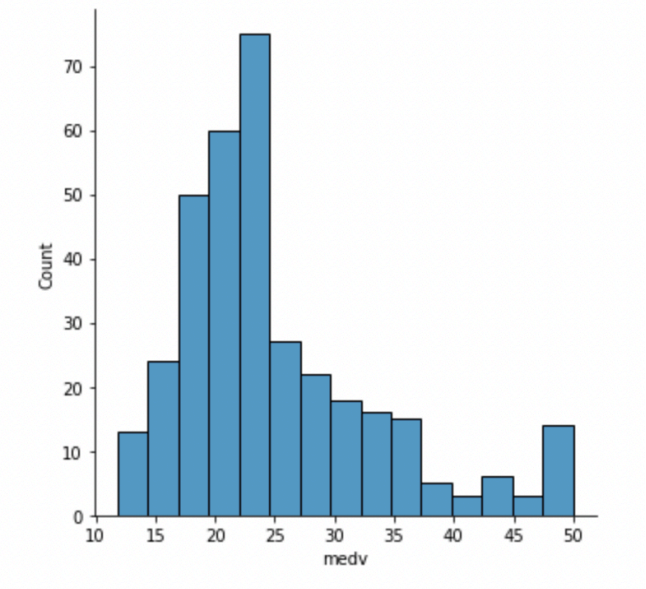
I would first get insights from the given data like finding correlation between different features and other necessary preprocessing and exploratory data analysis. I would then check if there any categorical features to do label encoding and then apply normalisation or standardisation to all features. Finally I would try various models and evaluate each one of them using different

evaluation metrics like root mean squared error, r2_score, etc and choose the one which gives the best predictions.

VISUALISATION

* Target feature

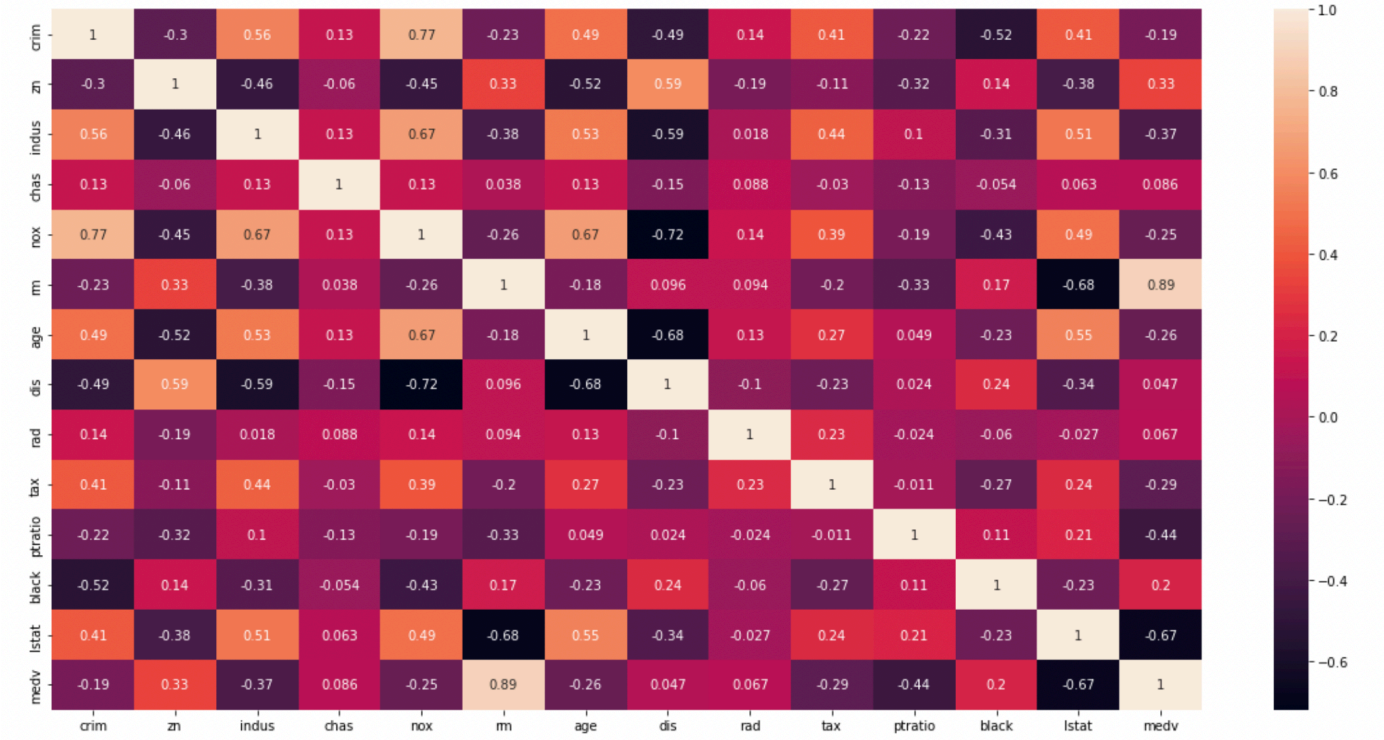
```
sns.displot(y);
```



* Correlation plot

```
plt.subplots(figsize = (20,10))
sns.heatmap(X.corr(),annot=True)
```

```
<AxesSubplot:>
```

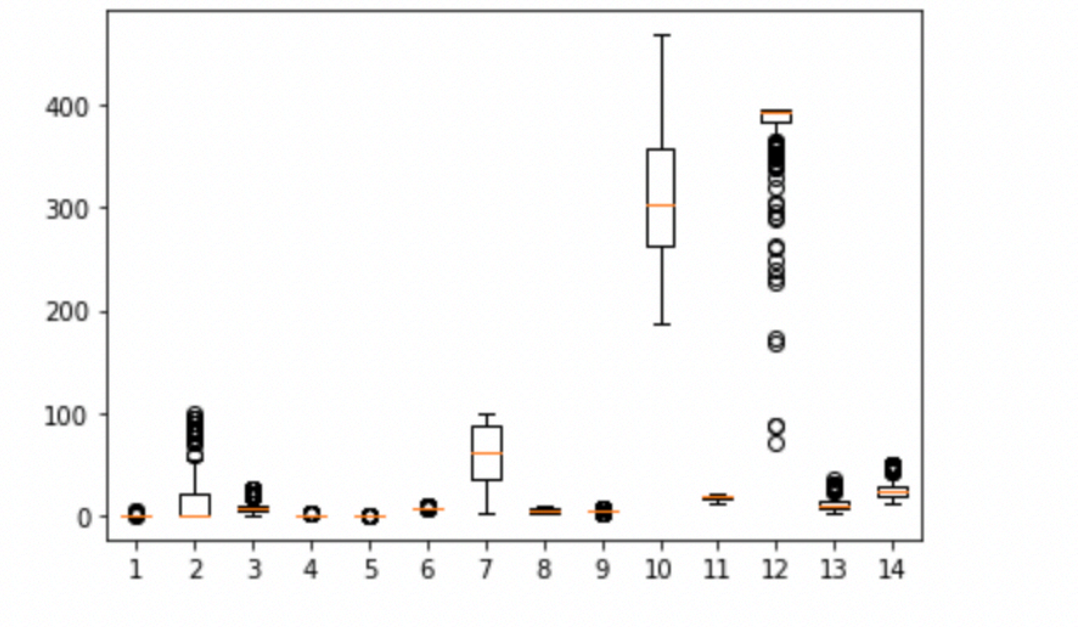


From this correlation plot it is evident that the feature “rm” has the highest correlation and other features like “lstat”, ”indus”, “ptratio” have decent correlations with target feature i.e “medv”.

* **Box plot**

This whisker plot displays a summary of a set of data containing the minimum, first quartile, median, third quartile, and maximum

```
plt.boxplot(X)
plt.show()
```



Before trying algorithms or models, I applied StandarScaler() to standardise the features resulting in zero mean ad one variance. Then dropped off target column(medv) from the dataset.

ALGORTHIMS

1) Linear Regression

I used to “rm”(average number of rooms per dwelling) for creating this model because of its high correlation value.

Evaluation metrics

```
Mean_squared_error=  173.35699725038413
Root_mean_squared_error=  13.166510443180613
```

```
r2_score(y_test,y_pred)
```

```
-1.6034174988186067
```

We get a negative r2_score, so this not the desired algorithm.

2) Multiple Linear Regression

I used the entire training set for this model.

Evaluation metrics

```
Mean_squared_error=  150.17914623231255
Root_mean_squared_error=  12.25476014584996
```

```
r2_score(y_test,y_pred)
```

```
-1.2553402715792283
```

r2_score is negative, so this is a bad model.

3) Polynomial Regression (best one)

Before applying I concatenated both the training and test dataset to a single data frame and then applied train_test split. This was done to ensure the data set is well shuffled and gives a positive r2_score.

I used the features “rm”, “ lstat”, “ptratio”, ”indus” for this model. I found out the root mean squared error and r2 score for each polynomial with degree ranging from 0 to 10 using “for loop”.

It turns out that the polynomial regression with a degree 3 gives the best scores.

Evaluation metrics

```
The best model is the model with a degree =  3
The Root mean squared error value =  5.06852608190581
```

```
r2_score(y_test, y_pred)
```

```
0.6516312958712601
```

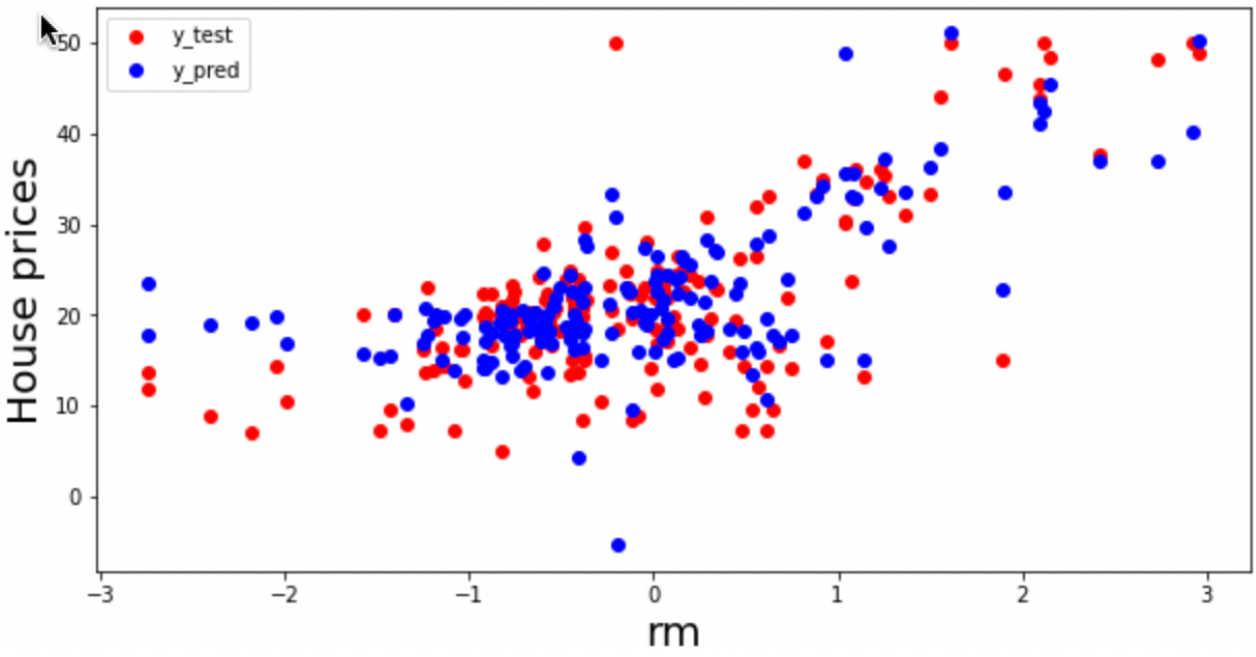
```
mean_absolute_error(y_test, y_pred)
```

```
4.085964385033607
```

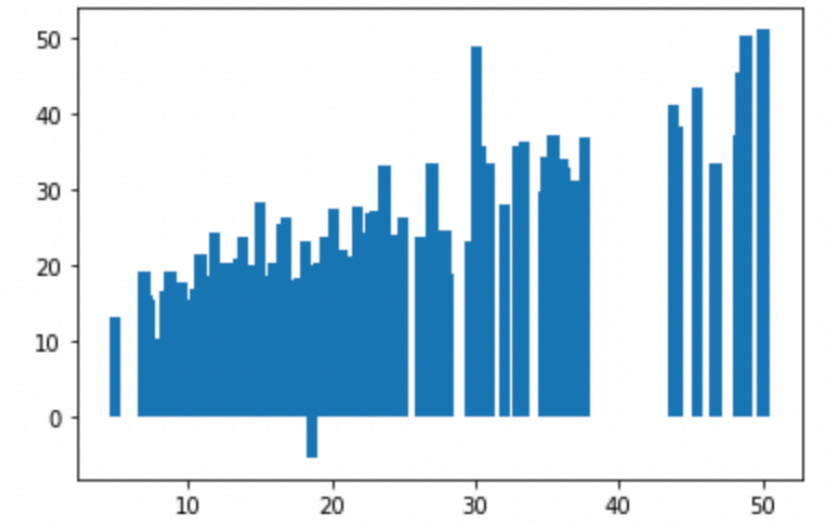
```
median_absolute_error(y_test, y_pred)
```

```
3.2819162132613773
```

Plots



```
barlist = plt.bar(y_test, y_pred)
plt.show()
```



* I tried another algorithm Random Forest Regressor which gave a lesser root mean squared error value and good r2_score.

CONCLUSION

Polynomial Regression of degree 3 using specific features or columns ('rm', 'lstat', 'indus', 'ptratio') gives the best prediction scores.

REFRENCES

1. Kaggle
2. Coursera ML course by Andrew NG