House Price Prediction using Machine Learning

Prajjwal Chittori (2K18/CO/249)
Pranav Roy (2K18/CO/250)
Dept. of Computer Science and Engineering)

Problems faced while buying a house

Problems faced during buying a house:

1) Buyers are generally not aware of factors that influence the house prices, Such as-

   i) No. of storeys.

   ii) Area outside the house

   iii) Geographic location and connectivity

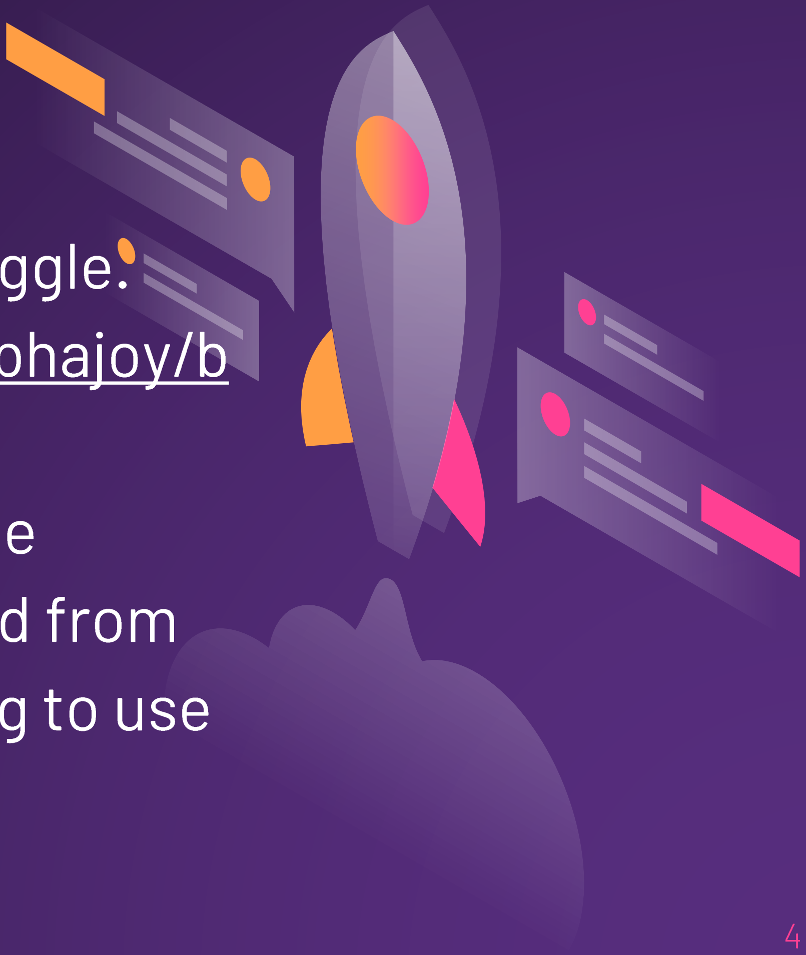
2) Hence real estate agents are trusted with the communication between buyers and sellers as well as laying down a legal contract for the transfer. This just creates a middle man and increases the cost of houses.

Agenda

- Our goal for this project was to use regression and classification techniques in order to estimate the sale price of a house in Bengaluru, India given the feature and pricing data for around 21,000 houses sold within one year.

Dataset

- Th e Dataset is acquired from Kaggle.
- (https://www.kaggle.com/amitabhajoy/bengaluru-house-price-data)
- There are 10 total attributes in the dataset, four of which we derived from current columns. We are planning to use all 10 attributes in our models.



TOOLS USED

python™

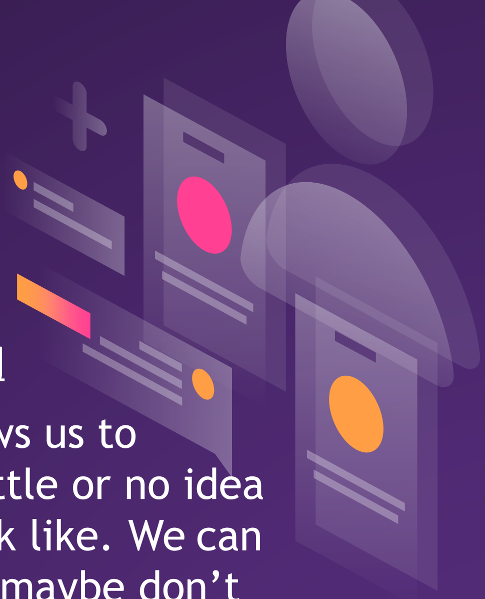
TensorFlow

ANACONDA®

jupyter

CO

Machine Learning Classification



Supervised Learning

In supervised learning, we are given a data set and already know what our correct output should look like, having the idea that there is a relationship between the input and the output. Supervised learning problems are categorized into "regression" and "classification" problems.

Unsupervised Learning

Unsupervised learning allows us to approach problems with little or no idea what our results should look like. We can derive structure where we maybe don’t know effect of variables by clustering the data based on relationships among the variables in the data. With unsupervised learning there is no feedback based on the prediction results

STEPS

Data Collection

First step is to collect data from different sources and then merge them to form the training dataset.

Source-Kaggle

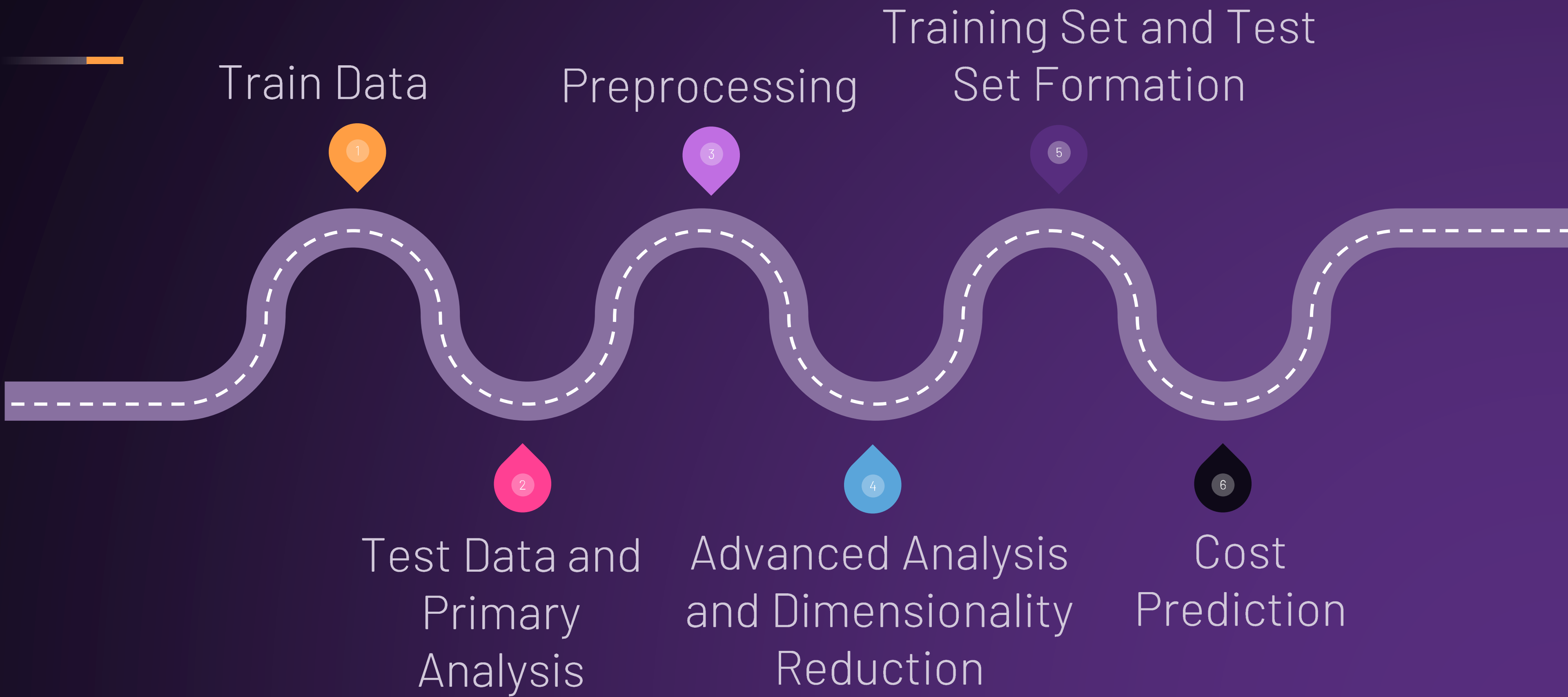
Training the model

We Use Multiple Linear Regression to train the Dataset.Multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable.

Cost Prediction

Based on the generated graphs and trained data we will predict the cost of the house.

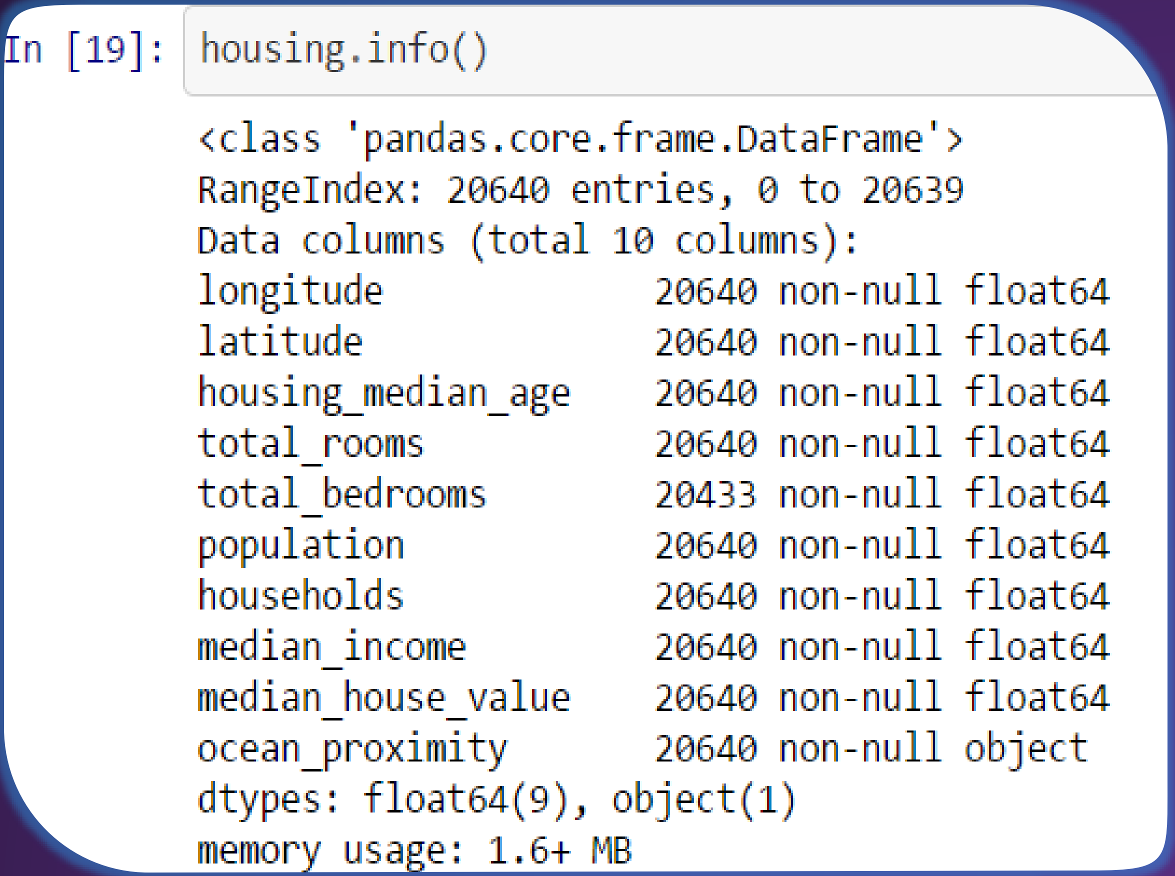
Architechtural Roadmap


Train Data
1
Test Data and Primary Analysis
2
Preprocessing
3
Advanced Analysis and Dimensionality Reduction
4
Training Set and Test Set Formation
5
Cost Prediction
6

Multiple Linear Regression

- Iteration
  - *Select one variable at a time from the variable importance table created using random forest*
- Significance
  - *Check significance of new variable along with existing variables by its t-value and probability of t- statistics.*
  - *If R-square is improved, keep the variable, else drop it*
- Multicollinearity
  - *Multicollinearity is checked at each step – ensuring the maximum value is < 4*
  - *If new variable has multicollinearity above threshold value – drop it*
  - *If introducing the new variable increases the multicollinearity of any existing variable – then the variable with lowest t-value is dropped*
- Model Accuracy
  - *After adding the new variable, the model accuracy on train and test data using error rate and MAPE is checked.*
  - *Drop the new variable if the model accuracy falls.*

Dataset Details



```
In [19]: housing.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
longitude             20640 non-null float64
latitude              20640 non-null float64
housing_median_age    20640 non-null float64
total_rooms           20640 non-null float64
total_bedrooms        20433 non-null float64
population            20640 non-null float64
households            20640 non-null float64
median_income         20640 non-null float64
median_house_value    20640 non-null float64
ocean_proximity       20640 non-null object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

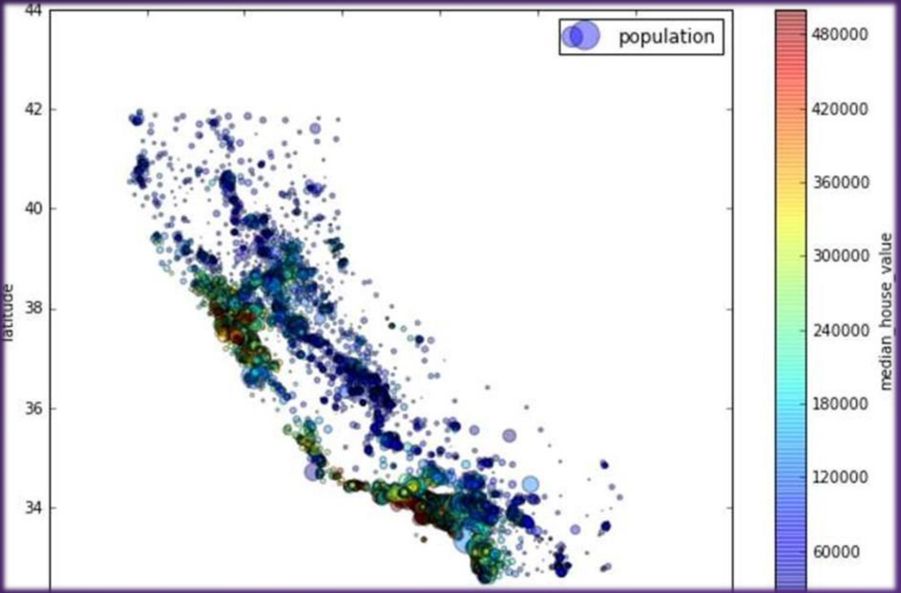
Visualizing using Plot Analysis

Exploratory Plot Analysis

Train Validate Test

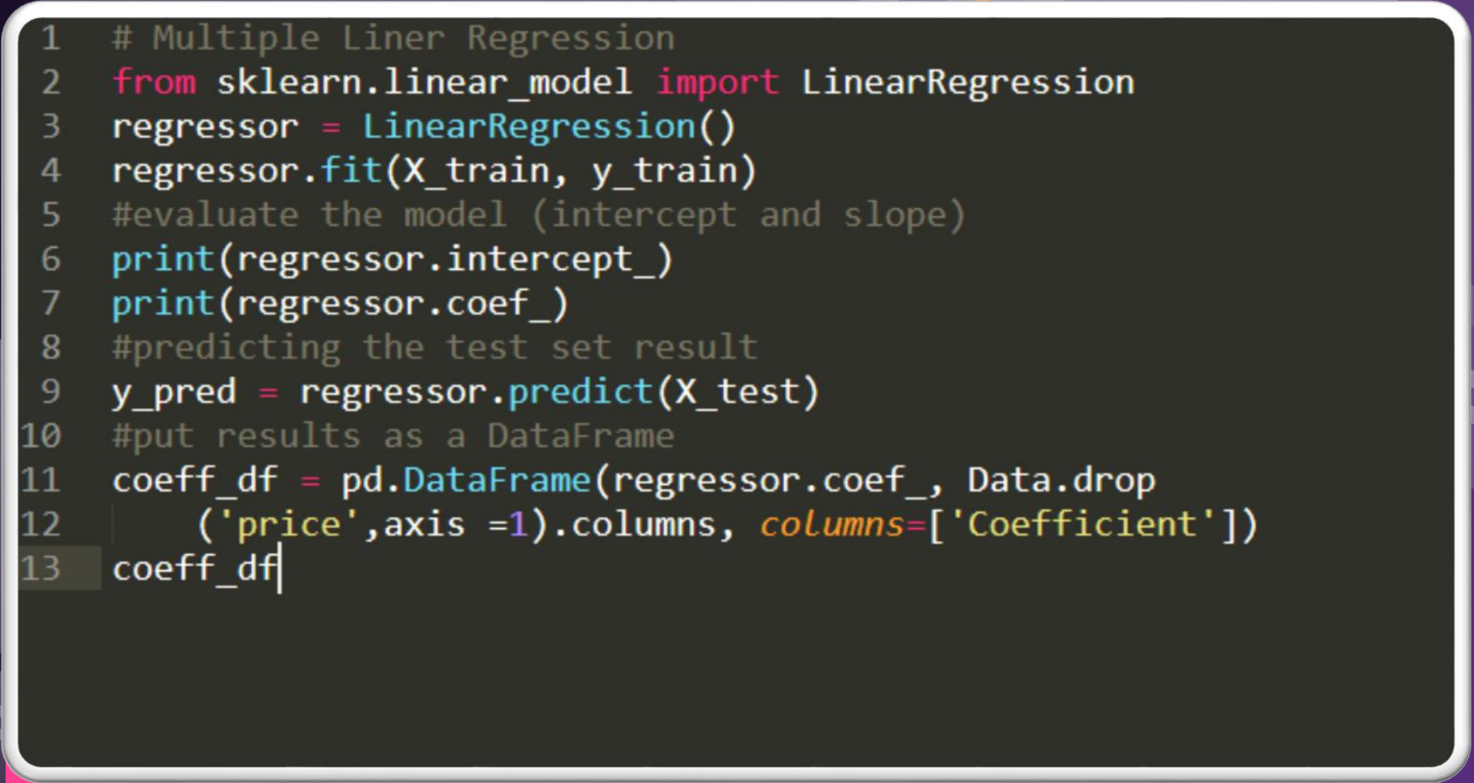
Correlation Plots

Median Analysis



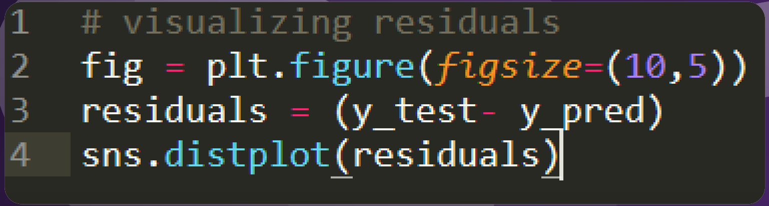
Plot to visualize role of latitude, longitude & population on the price of the house

Training Dataset Code

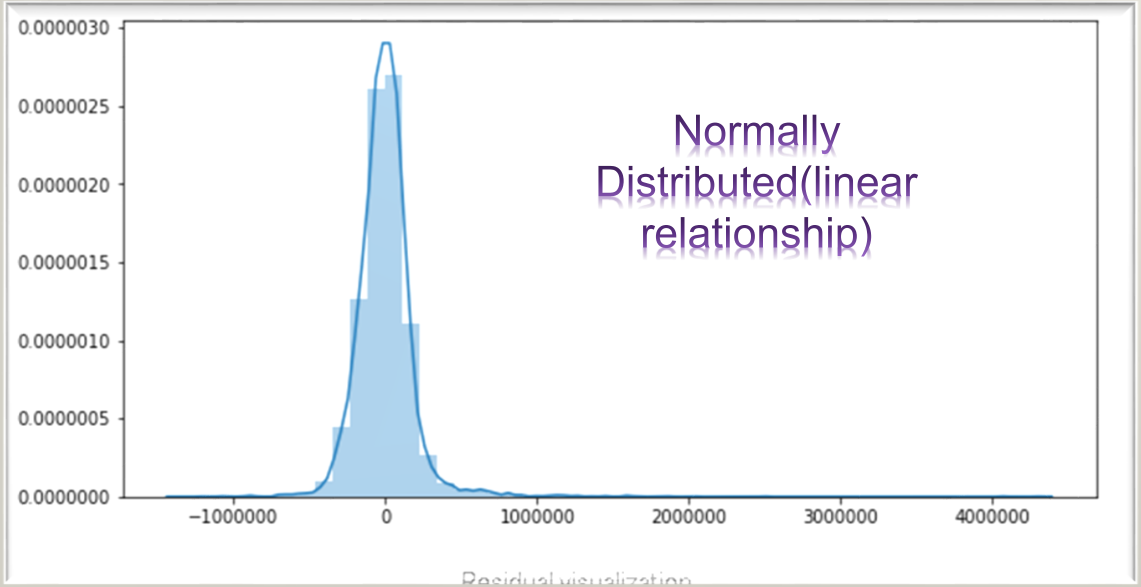


```python
# Multiple Liner Regression
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
#evaluate the model (intercept and slope)
print(regressor.intercept_)
print(regressor.coef_)
#predicting the test set result
y_pred = regressor.predict(X_test)
#put results as a DataFrame
coeff_df = pd.DataFrame(regressor.coef_, Data.drop
    ('price',axis =1).columns, columns=['Coefficient'])
coeff_df
```

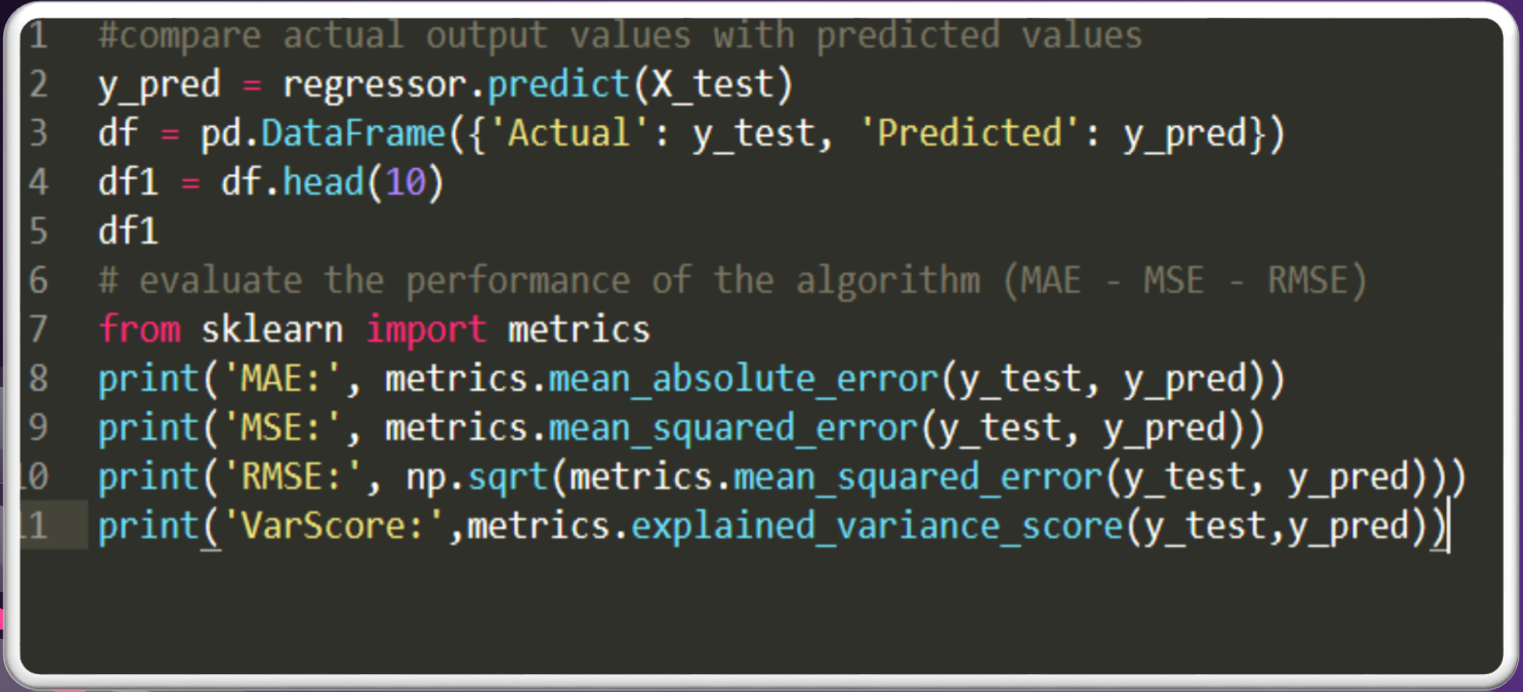
Visualizing Residuals



```
# visualizing residuals
fig = plt.figure(figsize=(10,5))
residuals = (y_test- y_pred)
sns.distplot(residuals)
```

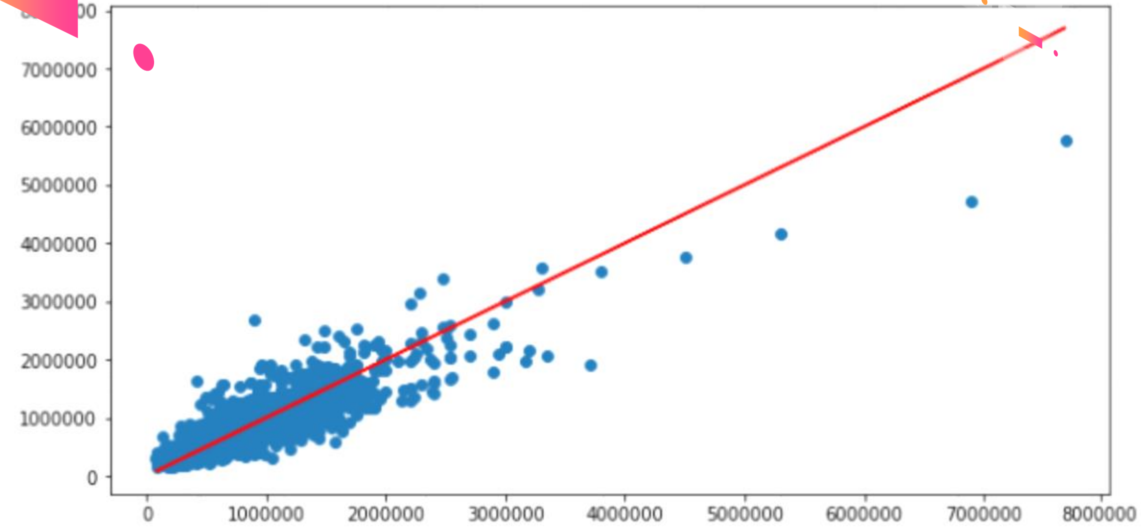


Comparing with Actual Results (Code)



```
#compare actual output values with predicted values
y_pred = regressor.predict(X_test)
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df1 = df.head(10)
df1
# evaluate the performance of the algorithm (MAE - MSE - RMSE)
from sklearn import metrics
print('MAE:', metrics.mean_absolute_error(y_test, y_pred))
print('MSE:', metrics.mean_squared_error(y_test, y_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
print('VarScore:',metrics.explained_variance_score(y_test,y_pred))
```

```
Mean Absolute Error: 96667.89
Mean Squared Error: 24912134897.75
Root Mean Squared Error: 157835.78
Variance score is: 0.81
```



Expected Results

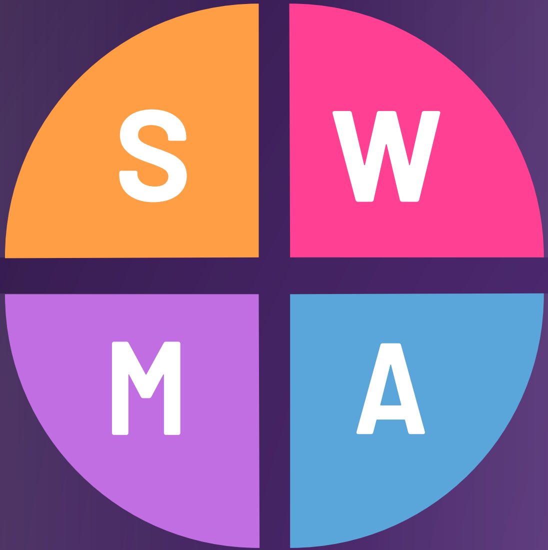
PERFORMANCE ANALYSIS

STRENGTHS

Results are expected to be fairly accurate with accuracy rate of more than 70%

WEAKNESSES

Model can improved further with more time given for Training Dataset and using Keras regression



Results can be further improved by removing Multi-colinearity

MULTI COLINEARITY

Autocorrelation refers to the degree of correlation between the values of the same variables across different observations in the data.

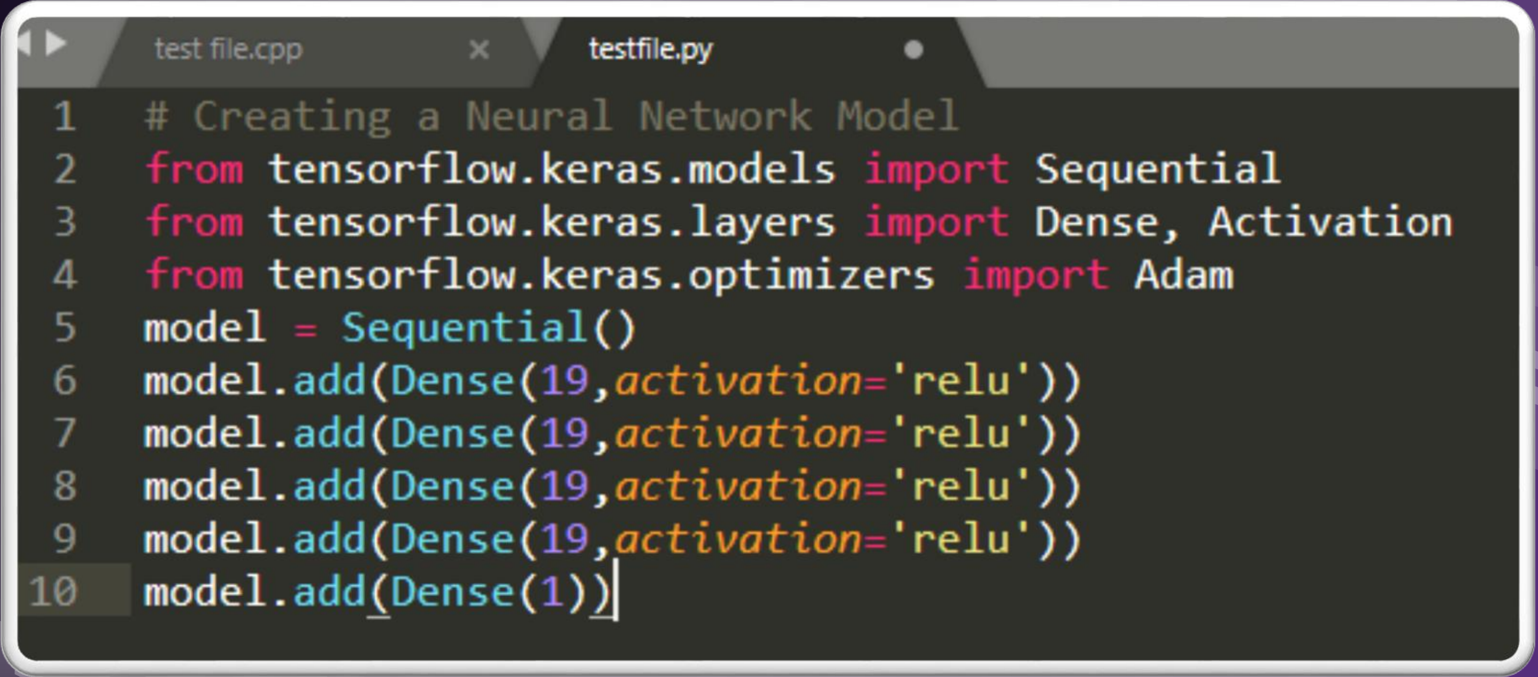
AUTOCORRELATION

Keras Regression

Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.

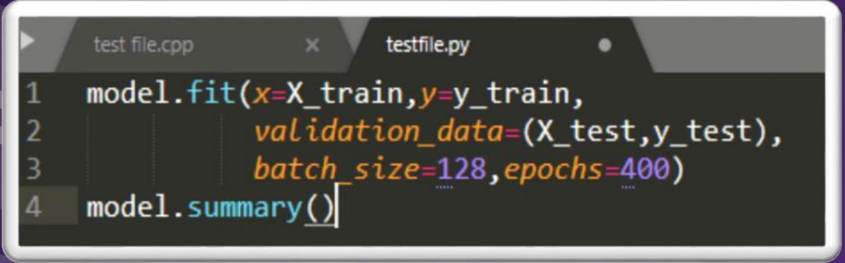
We can easily fit the regression data with Keras sequential model and predict the test data. Preparing data, Defining the model, Fitting with **Keras Regressor** (accuracy check and visualizing the results) Fitting with the sequential model (accuracy check and visualizing the results can all be done with Keras

Training Dataset Code

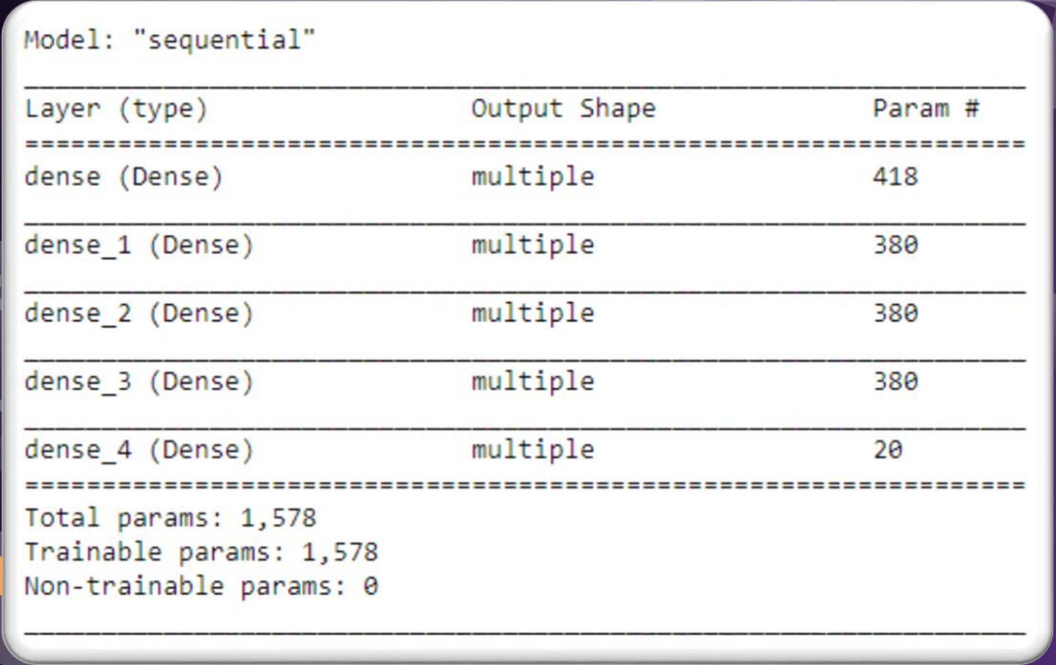


```python
# Creating a Neural Network Model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.optimizers import Adam
model = Sequential()
model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))
model.add(Dense(1))
```

Training

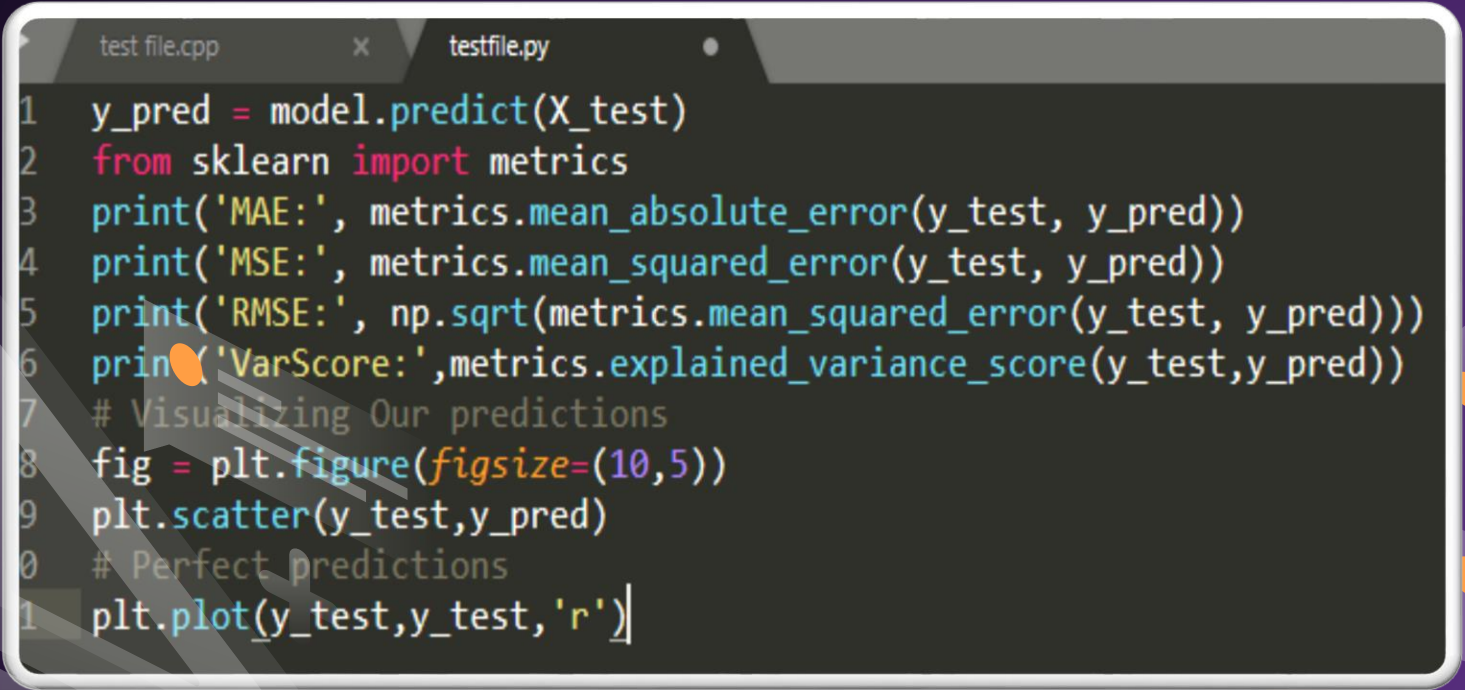


```
model.fit(x=X_train,y=y_train,
          validation_data=(X_test,y_test),
          batch_size=128,epochs=400)
model.summary()
```



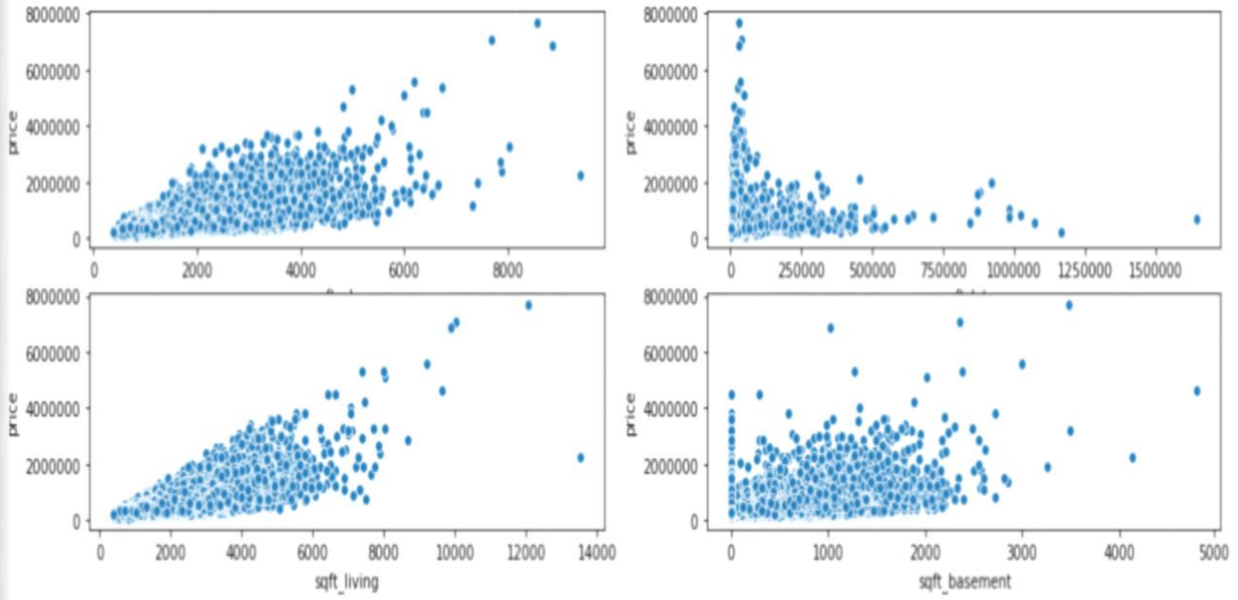
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                multiple                  418
_________________________________________________________________
dense_1 (Dense)              multiple                  380
_________________________________________________________________
dense_2 (Dense)              multiple                  380
_________________________________________________________________
dense_3 (Dense)              multiple                  380
_________________________________________________________________
dense_4 (Dense)              multiple                  20
=================================================================
Total params: 1,578
Trainable params: 1,578
Non-trainable params: 0
_________________________________________________________________
```

Comparing with Actual Results (Code)

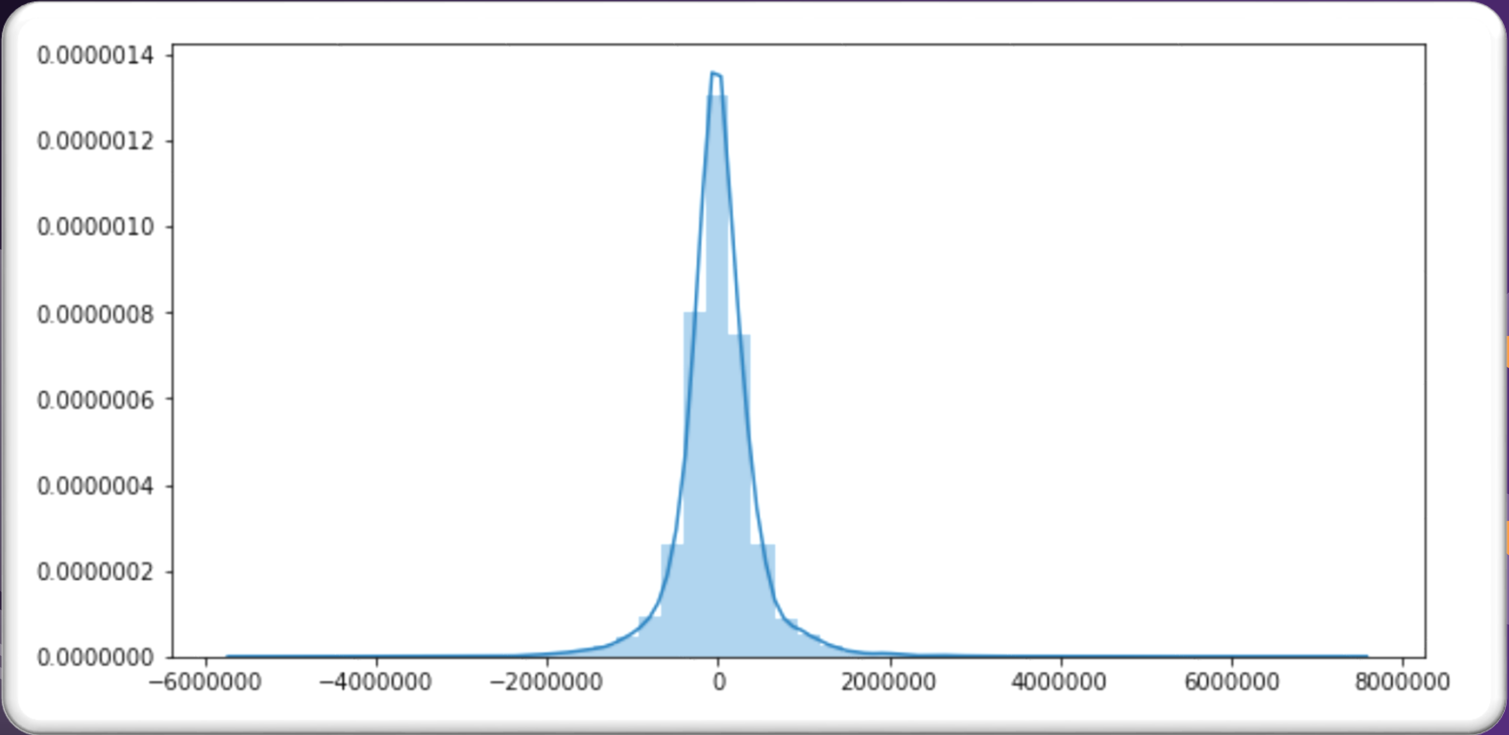


```
y_pred = model.predict(X_test)
from sklearn import metrics
print('MAE:', metrics.mean_absolute_error(y_test, y_pred))
print('MSE:', metrics.mean_squared_error(y_test, y_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
print('VarScore:',metrics.explained_variance_score(y_test,y_pred))
# Visualizing Our predictions
fig = plt.figure(figsize=(10,5))
plt.scatter(y_test,y_pred)
# Perfect predictions
plt.plot(y_test,y_test,'r')
```

Expected Results



Visualising Residuals



THANKS!