Linear Regression using Sklearn

This note book cover the following concepts

1. Visualization

2. Sea born

1. Install the following modules
   1. Numpy
   2. Pandas
   3. Matplotlib.pyplot
   4. %matplotlib inline
   5. Sklearn
   6. Seaborn
   7. Warnings with its filter

## **TRAINING DATA PRE-PROCESSING**

The first step in the machine learning pipeline is to clean and transform the training data into a useable format for analysis and modeling.

* + 1. As such, data pre-processing addresses:
    2. Assumptions about data shape
    3. Incorrect data types
    4. Outliers or errors
    5. Missing values
    6. Categorical variables

1. Import USA Housing data
2. Check Data Shape, after loading the dataset, I examine its shape to get a better sense of the data and the information it contains.
3. View first rows
4. View Data Info
5. Missing Data, A heatmap will help better visualize what features as missing the most information.
6. Remove Address features
7. Remove rows with missing data
8. Check data
9. Numeric summary of dataframe

# **GETTING MODEL READY**

Now that we've explored the data, it is time to get these features 'model ready'. Categorial features will need to be converted into 'dummy variables', otherwise a machine learning algorithm will not be able to take in those features as inputs.

1. Shape of train data

Now the train data is perfect for a machine learning algorithm:

* all the data is numeric
* everything is concatenated together

## **OBJECTIVE 2: MACHINE LEARNING**

Next, I will feed these features into various classification algorithms to determine the best performance using a simple framework: \*\***Split, Fit, Predict, Score It**.\*\*

## **Target Variable Splitting**

We will spilt the Full dataset into \*\***Input**\*\* and \*\***target**\*\* variables

* Input is also called \*\***Feature Variables**\*\*
* Output refers to Target \*\***variables**\*\*

1. Split data to be used in the models, Create matrix of features, x grabs everything else but ‘Price’
2. Create target variable, y is the column we're trying to predict
3. x Represents the Features and its shape
4. y Represents the Target / outcome and its shape
5. using sklearn with preprocessing class and create instance to of preprocessing class and its StandardScaler method and fit method with params x and save in pre\_process variable.
6. Now, use x\_transform with perform fit\_transform input data i.e. x.
7. Use x\_transform and y\_transform variables to split the training data into train and test set, using sklearn.model\_selection and impot train\_test\_split

## **LINEAR REGRESSION**

### **Model Training**

1. For Fit, the following modules import:
   1. sklearn.linear\_model import LinearRegression
   2. sklearn.pipline import make\_pipeline
   3. sklearn.preprocessing import StandardScaler
2. Create instance of model with name of lin\_reg
3. Pass training data into model with fit method and pass train data of x and y.

*LinearRegression()*

***In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.  
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.***

## **Model Testing**

### **Class prediction**

1. Predict with name y\_pred of x\_test, and print its shape and data
2. Using Seaborn View scatterplot with x is y\_test, y is y\_pred, color, blue, label is actual Data point.
3. Using matplotlib to show plot two array inside min and max y\_test data, and color is red and label is Ideal line, and then plt with legend method also show method.
4. Combine actual and predicted values side by side and store in variable result with using np.column\_stack and params y\_test, y\_pred.

Printing the results

print("Actual Values | Predicted Values")

print("-----------------------------")

print actual and predicted values using for loop from results.

## **Residual Analysis**

Residual analysis in linear regression is a way to check how well the model fits the data. It involves looking at the differences (residuals) between the actual data points and the predictions from the model.

In a good model, the residuals should be randomly scattered around zero on a plot. If there are patterns or a fan-like shape, it suggests the model may not be the best fit. Outliers, points far from the others, can also affect the model.

Residual analysis helps ensure the model's accuracy and whether it meets the assumptions of linear regression. If issues are found, adjustments to the model may be needed to improve its performance.

1. Find out residual value i.e. deduct y\_pred.reshape(-1) from actual value and print it.
2. Show Distribution plot for Residual (difference between actual and predicted values)

*(It represents that our mode is not skewed as the distribution is center aligned but note the values of the X and Y axis they in power of 6. Which means the difference between actual and predicted value is high)*

## **Model Evaluation**

**Linear Regression**

1. Score it using sklearn.metrics with import mean\_squared\_error model

print('Linear Regression Model')

# Results

print('--'\*30)

1. To find out mse(mean squared error) with its name method
2. To find out rmse(root mean squared error) using np.sqrt
3. Print evaluation metrics

print("Mean Squared Error:", )

print("Root Mean Squared Error:", )

*(Previous: Linear Regression Model*

*Mean Squared Error: 10100187858.864885*

*Root Mean Squared Error: 100499.69083964829*

*10170939558)*

1. Match previous and current Model evaluation and check previous and current Linear Regression Model difference.
2. Check its shape

## **Decision Tree**

1. For decision tree using following modules:
   1. sklearn.tree import DecisionTreeRegressor
   2. sklearn.ensemble import RandomForestRegressor
2. To initialize Decision tree with its instance namely rf\_regressor.
3. Fit to train data.
4. Predicting the Sale Prices using test set in x\_test.
5. To find out and save in the variable Dtr and use mean squared error method with y\_pred and y\_test params
6. Print Decision Tree Regression Accuracy with test set.
7. For Random Forest tree using following modules:
   1. sklearn.tree import DecisionTreeRegressor
   2. sklearn.ensemble import RandomForestRegressor
8. To initialize Decision tree with its instance namely rf\_regressor.
9. Fit to train data i.e. x\_train and y\_train.
10. Predicting the Sale Prices using test set in x\_test.
11. To find out and save in the variable RFr and use mean squared error method with y\_pred and y\_test params
12. Print Random Forest Regression Accuracy with test set.
13. For Gradient Boosting Regression using following modules:
    1. sklearn.tree import DecisionTreeRegressor
    2. sklearn.ensemble import RandomForestRegressor
    3. sklearn.ensemble import GradientBoostingRegressor
14. To initialize Decision tree with its instance namely rf\_regressor.
15. Fit to train data i.e. x\_train and y\_train.
16. Predicting the Sale Prices using test set in x\_test.
17. To find out and save in the variable GBr and use mean squared error method with y\_pred and y\_test params
18. Print Random Forest Regression Accuracy with test set.
19. Sample model scores (replace these with your actual model scores)
    1. Create dictionary with the name of model\_scores and save the all-Model answers.
    2. Sort the model scores and save in the variable sorted\_scores with in ascending order based on their values (lower values first)
    3. Display the ranking of the models