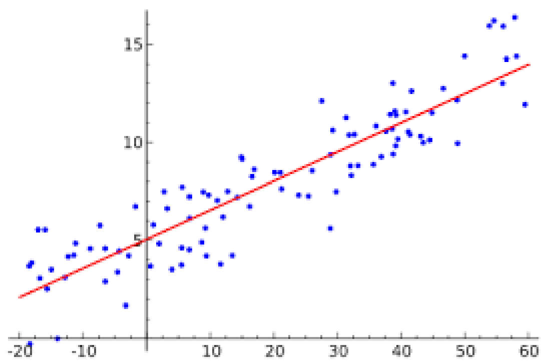
Assignment 1

Nisarg Negi
MS in Data Science
University at Buffalo, Buffalo, NY 14260
nisargne@bufallo.edu

1 Simple Linear Regression

Regression models are used to predict a dependent variable using other known (denoted by Y) or independent variables (denoted by X). Linear regression is one such regression model in which prediction is made using one independent variable(X) to predict a dependent variable(Y). The relationship between X & Y is assumed to be linear.

In Linear regression, we try to find the best fitting line to minimize the errors in prediction. This line is called the "line of best fit" i.e. the line of regression on which the errors will be minimal. We try to minimize the distance between the observed value and the predicted value. So, our general equation for Linear regression comes out to be:



$$Y = b_0 + b_1X_1$$

Where,

Y = Dependent variable

b_0 = Y-intercept

b_1 = slope of the line

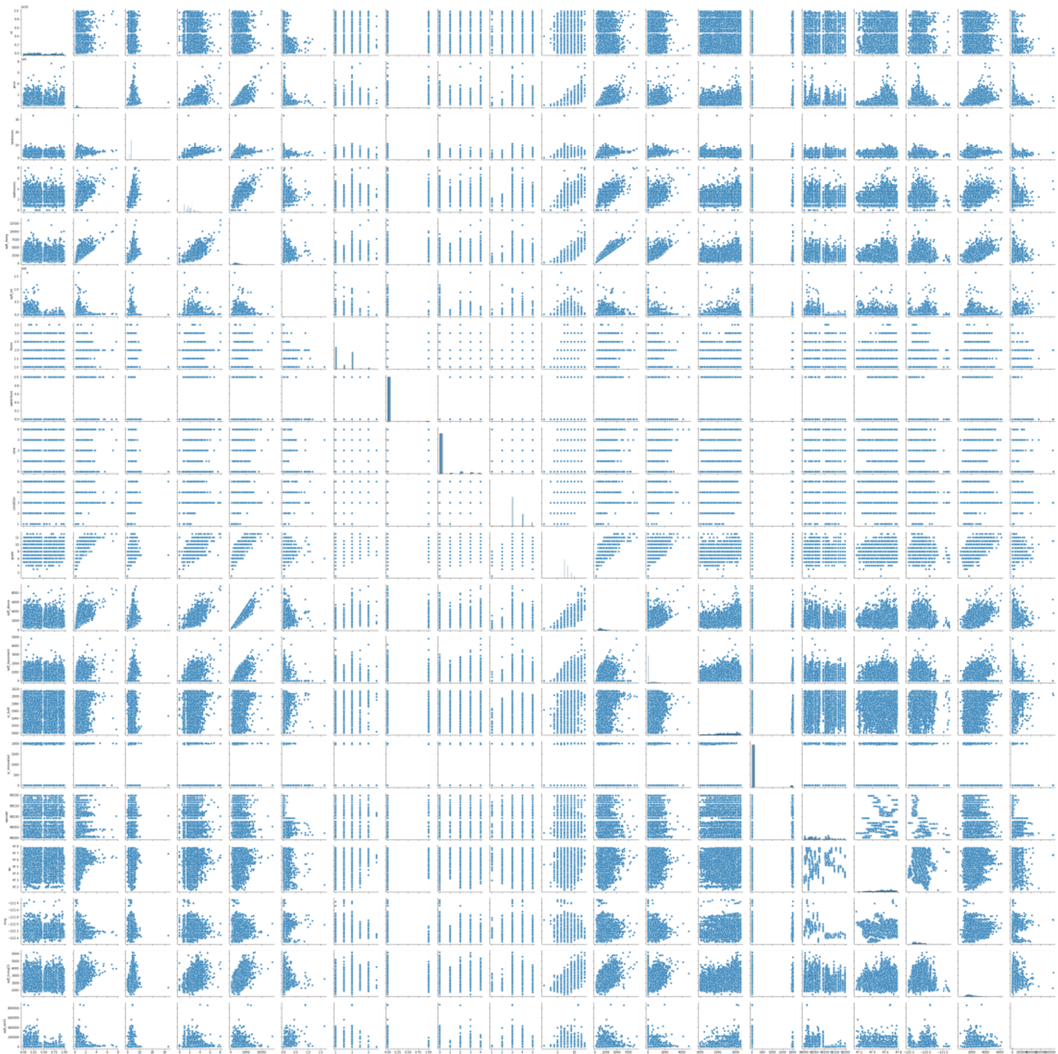
x_1 = independent variable

1.1 Experiment

We are going to use a housing dataset [1] to illustrate the application and workings of Simple Linear Regression. We will use this dataset and train a model to predict housing prices using linear regression. The data has features as below:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             21613 non-null  int64  
 1   date           21613 non-null  object 
 2   price          21613 non-null  float64
 3   bedrooms       21613 non-null  int64  
 4   bathrooms      21613 non-null  float64
 5   sqft_living    21613 non-null  int64  
 6   sqft_lot       21613 non-null  int64  
 7   floors         21613 non-null  float64
 8   waterfront     21613 non-null  int64  
 9   view           21613 non-null  int64  
 10  condition      21613 non-null  int64  
 11  grade          21613 non-null  int64  
 12  sqft_above     21613 non-null  int64  
 13  sqft_basement  21613 non-null  int64  
 14  yr_built       21613 non-null  int64  
 15  yr_renovated   21613 non-null  int64  
 16  zipcode        21613 non-null  int64  
 17  lat            21613 non-null  float64
 18  long           21613 non-null  float64
 19  sqft_living15  21613 non-null  int64  
 20  sqft_lot15     21613 non-null  int64  
dtypes: float64(5), int64(15), object(1)
memory usage: 3.5+ MB
```

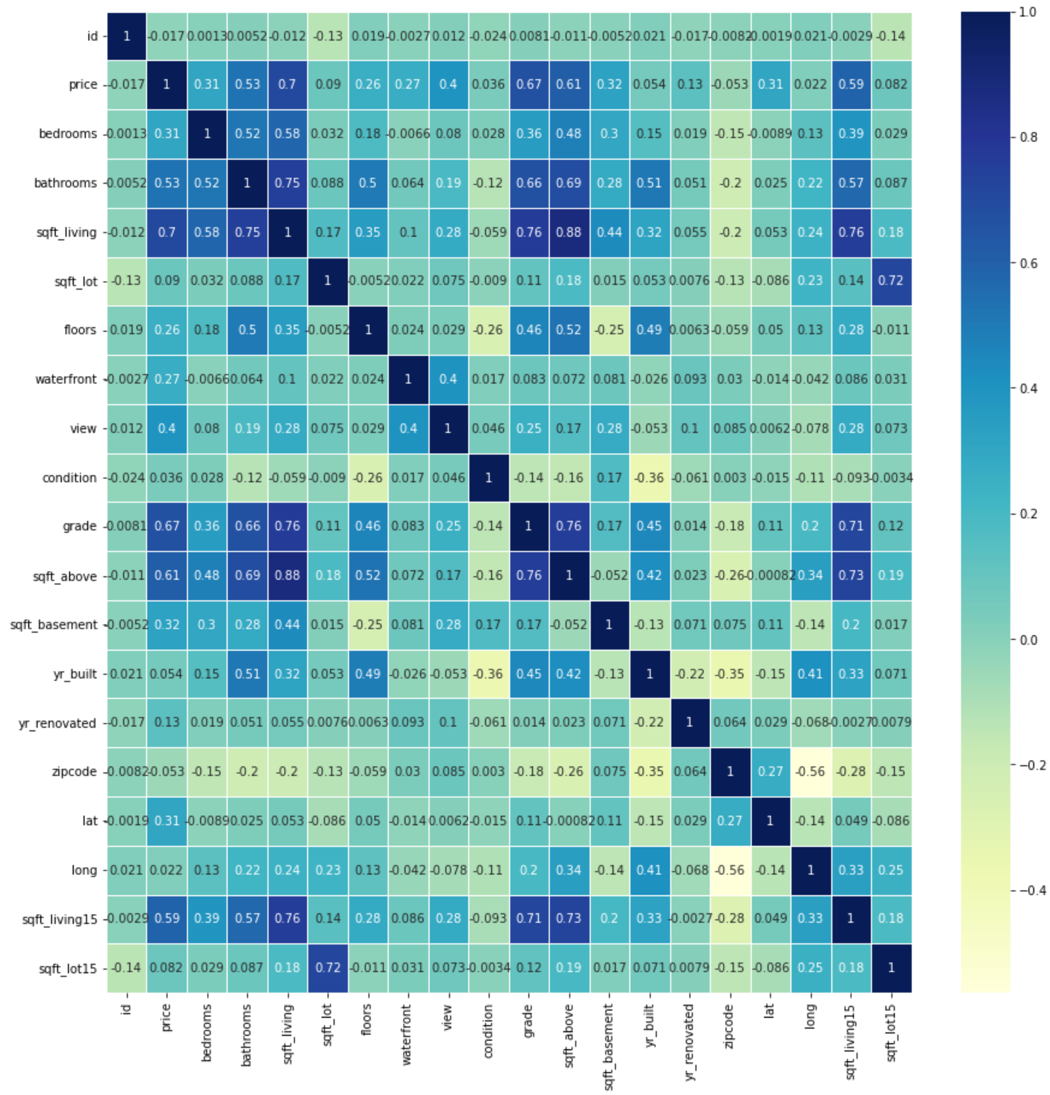
1.2 Feature Selection



Above we built an sns plot in which the diagonal plots show the distribution of a single variable and the other plots in the upper and lower triangle show the relationship between two variables. From the above we can tell that for our target variable (Y) the distributions that are most linear are:

-sqft_living

-sqft_above



Next, we will create a correlation plot. Correlation helps us understand the relation between two variables. It ranges from -1 to +1. If the value of the relationship is zero, there is no correlation. As the value of correlation changes from zero to a positive or negative one, the linear relationship grows stronger. We can observe from the above plot that the best correlation for the price variable is from:

-sqft_living

-grade

-sqft_above

As sqft_living is having the best correlation with price we choose this variable for our analysis.

1.3 Training the model

We will then store the price variables(Y) and sqft_living(X) and split this data into training and test in the ratio of 0.75 and .25 respectively. The train dataset is used for training our simple linear regression model.

Now we will train to fit our model on the train dataset such and test the model to predict using the Test Dataset. We will then compare the Actual values to the predicted values:

	Actual	Prediction
0	297000.0	359236.95
1	1578000.0	1267349.35
2	562100.0	362039.77
3	631500.0	275152.47
4	780000.0	849729.76
...	...	...
5399	649990.0	555434.08
5400	390000.0	241518.68
5401	774950.0	914194.53
5402	372500.0	289166.55
5403	599995.0	412490.46

We can also find the intercept and coefficient values from the trained model:

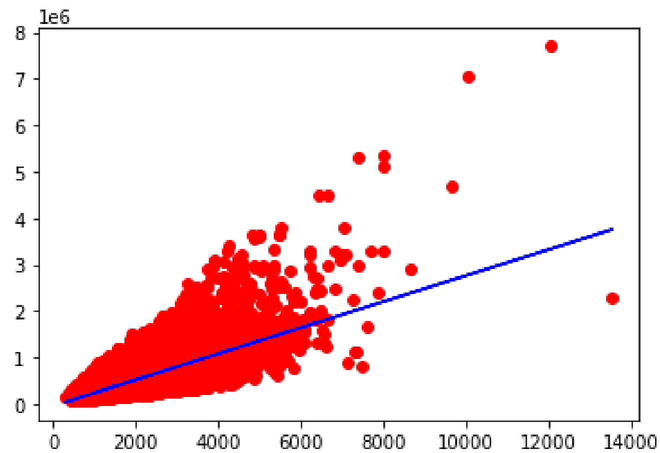
```
-41565.741905973875
[280.28160476]
```

From this we can build our equation with intercept value = -41565.741905973875 and coefficient value = 280.28160476. So, our linear regression equation looks like:

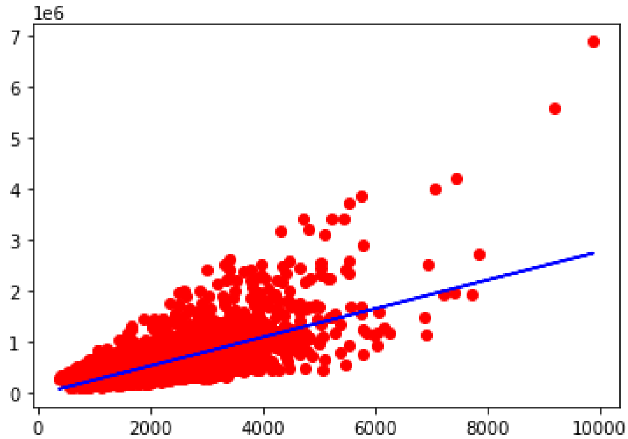
$$Y = 280.28X - 41565.74$$

We will now perform visualization our results:

-train dataset using actual data and predicted data:



- test dataset using actual data and predicted data:



1.4 Validation

We will use mean absolute error for validation. MAE calculates the average error between the actual and the predicted values. MAE is:

```
MAE:  173359.02468191707
```

This means that on average our predictions we off by 173359.02 which means that our model performs poorly.

References

[1] Housing Price dataset from https://www.kaggle.com/datasets/shivachandel/kc-house-data

[2] https://medium.com/analytics-vidhya/simple-linear-regression-using-python-98ddd7e6b391

2 Multiple Linear Regression

Multiple linear Regression is based upon linear regression but it takes more than one independent variable to predict a dependent variable. It fits a linear equation based upon two or more dependent variables[X] to predict the independent variable [Y] and forms a line of best regression.

Thus the equation is:

$$Y = b_0 + b_1X_1 + b_2X_2 + b_3X_3 + \dots + b_nX_n$$

Where,

Y = Dependent variable

b_0 = Y-intercept

$b_1, b_2, b_3, \dots, b_n$ = slope of the line

$x_1, x_2, x_3, \dots, x_n$ = independent variable

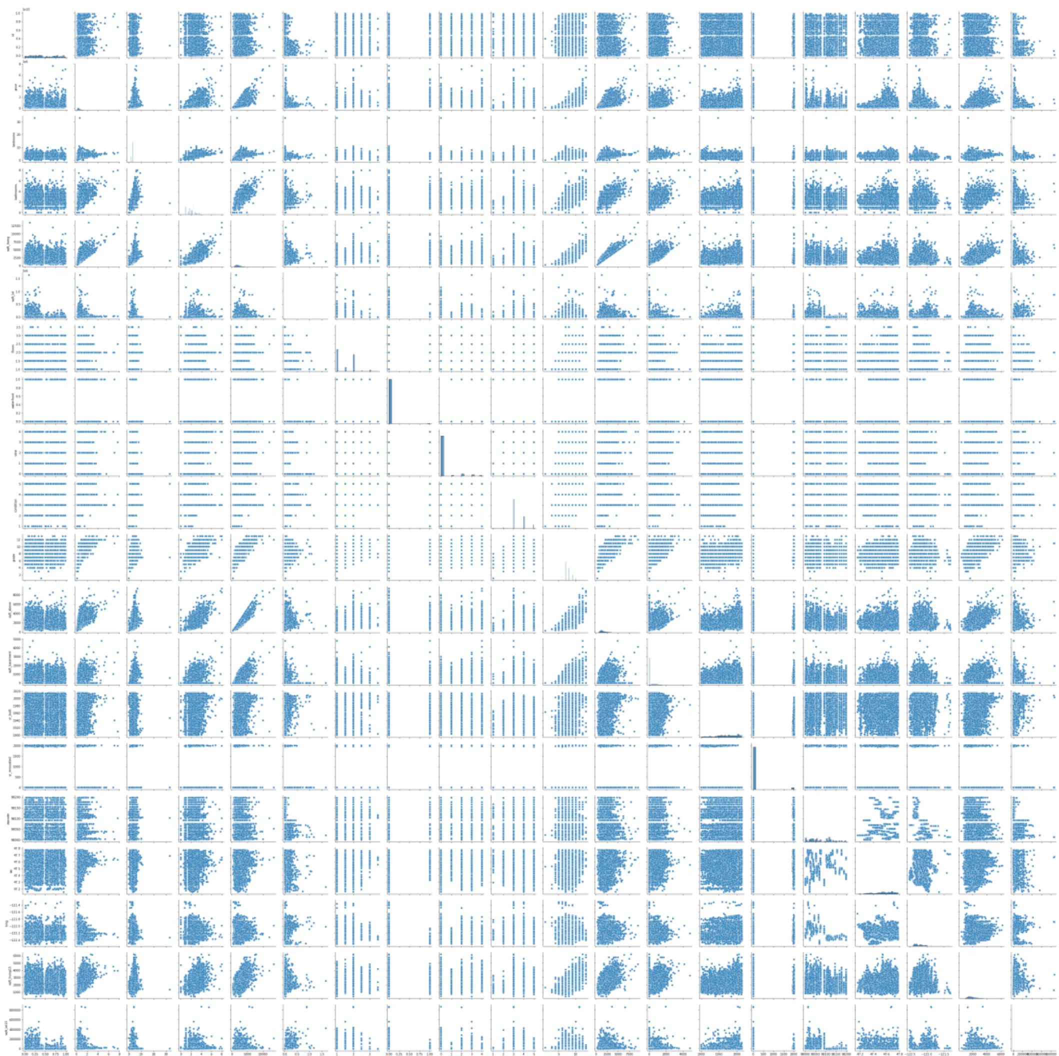
Example: Predicting if a customer will buy a car based on his salary, city, gender, etc

2.1 Experiment

We are going to use a housing dataset [1] to illustrate the application and workings of Multiple Linear Regression. We will use this dataset and train a model to predict housing prices using linear regression. The data has features as below:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             21613 non-null  int64
 1   date           21613 non-null  object
 2   price          21613 non-null  float64
 3   bedrooms       21613 non-null  int64
 4   bathrooms      21613 non-null  float64
 5   sqft_living    21613 non-null  int64
 6   sqft_lot       21613 non-null  int64
 7   floors         21613 non-null  float64
 8   waterfront     21613 non-null  int64
 9   view           21613 non-null  int64
 10  condition      21613 non-null  int64
 11  grade          21613 non-null  int64
 12  sqft_above     21613 non-null  int64
 13  sqft_basement  21613 non-null  int64
 14  yr_built       21613 non-null  int64
 15  yr_renovated   21613 non-null  int64
 16  zipcode        21613 non-null  int64
 17  lat            21613 non-null  float64
 18  long           21613 non-null  float64
 19  sqft_living15  21613 non-null  int64
 20  sqft_lot15     21613 non-null  int64
dtypes: float64(5), int64(15), object(1)
memory usage: 3.5+ MB
```

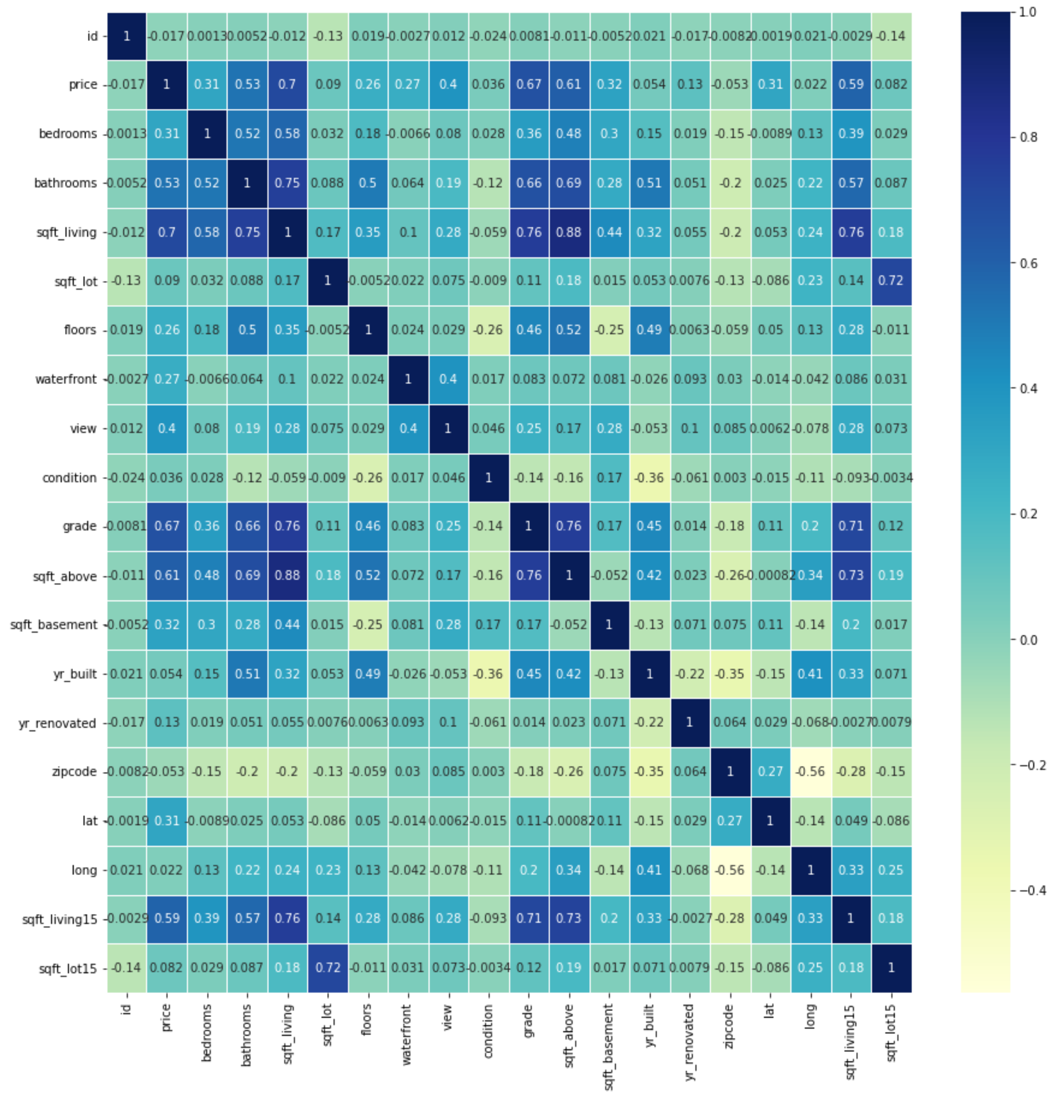
2.2 Feature Selection



Above we built an sns plot in which the diagonal plots show the distribution of a single variable and the other plots in the upper and lower triangle show the relationship between two variables. From the above we can tell that for our target variable (Y) the distributions that are most linear are:

-sqft_living

-sqft_above



Next, we will create a correlation plot. Correlation helps us understand the relation between two variables. It ranges from -1 to +1. If the value of the relationship is zero, there is no correlation. As the value of correlation changes from zero to a positive or negative one, the linear relationship grows stronger. We can observe from the above plot that the best correlation for the price variable is from:

-sqft_living

-grade

-sqft_above

2.3 Training the model

We will perform label encoding for all the features of our dataset. We will then store the price variables(Y) and all the other independent features(X) and split this data into training and test in the ratio of 0.75 and .25 respectively. The train dataset is used for training our simple linear regression model.

Now we will train to fit our model on the train dataset and test the model to predict using the Test Dataset. Predicted values are :

```
array([[678844.66978338],
       [678844.66978338],
       [316906.51667245],
       ...,
       [107870.92074046],
       [360686.45256026],
       [110865.05283179]])
```

We can also find the intercept and coefficient values from the trained model:

```
[11836332.32264203]
[[-3.39679112e+04  4.09513566e+04  1.08176103e+02  1.50042181e-01
  -8.57023413e+02  5.94541283e+05  5.21325064e+04  2.73430448e+04
   9.06339579e+04  7.32085991e+01  3.49675041e+01 -2.92310294e+03
   2.06021675e+01 -5.89671754e+02  5.82285879e+05 -1.91371357e+05
   3.61416693e+01 -4.41271499e-01]]
```

From this we can build our equation with intercept value = 11836332.32264203 and coefficient value = -3.39679112e+04, 4.09513566e+04, 1.08176103e+02….

2.4 Validation

We will use mean absolute error for validation. MAE calculates the average error between the actual and the predicted values. MAE is:

```
MAE:  131275.0952232879
```

This means that on average our predictions we off by 131275.09 which means that our model performs poorly but better that simple linear regression

References

[1] Housing Price dataset from https://www.kaggle.com/datasets/shivachandel/kc-house-data

[2] https://medium.com/machine-learning-with-python/multiple-linear-regression-implementation-in-python-2de9b303fc0c

3 Logistic Regression

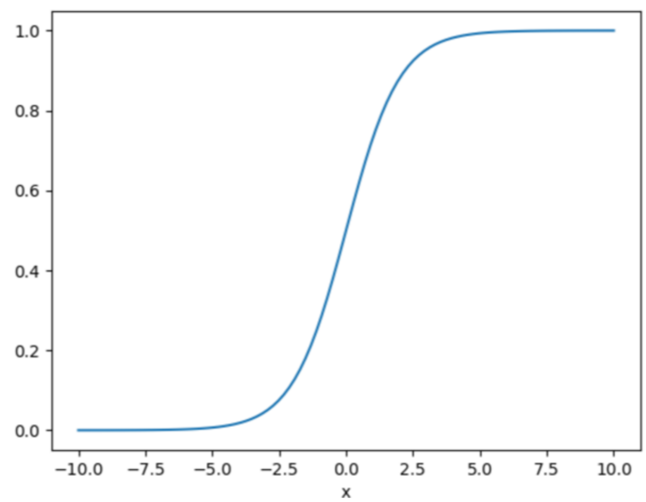
Logistic regression is a supervised machine learning algorithm classification that is used to predict categorical data. In logistic regression, the target variable or dependent variable(Y) is a binary variable and has either 1(true) or 0(false) as values. So essentially the model predicts P(Y=1) as a function of X. For example, will a certain customer buy a product or not?

Logistic regression uses a sigmoid function which is a mathematical function that takes any real number and returns the value as either 0 or 1.

Sigmoid function [2]:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Sigmoid graph



The sigmoid function forms an S shaped graph, so when x becomes infinity the value of P(Y) is 1, and when X reaches negative infinity the value of P(Y) is 0. The model sets a threshold as to above which value the model predicts if the event Y will happen or not. Example the thresh hold might be p(Y) = 0.5, any value above it will return true for event P happening and false for it not happening.

3.1 Experiment

We are going to use the dataset from the National Institute of Diabetes and Digestive and Kidney Diseases[1] to train a model to predict whether a patient has diabetes or not. The data has features as below:

```
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

3.2 Feature Selection

We will create a correlation plot. Correlation helps us understand the relation between two variables. It ranges from -1 to +1. If the value of the relationship is zero, there is no correlation. As the value of correlation changes from zero to a positive or negative one, the linear relationship grows stronger. We can observe from the above plot that the best correlation for the outcome variable is from:

-glucose

-Insulin

As glucose is having the best correlation with outcome we choose this variable for our analysis.

3.3 Training the model

We will then store the outcome variable(Y) and Insulin(X), and split this data into training and test in the ratio of 0.75 and .25 respectively. The train dataset is used for training our logistic regression model.

We will use the standard scaler in scikit-learn to normalize the independent variable as logistic regression uses gradient descent and if some features have higher magnitude and some have lower magnitude the convergence becomes difficult. Scaling helps make convergence happen faster.

Now we will train to fit our model on the train dataset and test the model to predict using the Test Dataset.

	Actual	Prediction
0	1	0
1	1	0
2	0	0
3	0	0
4	1	1
...	...	...
226	1	1
227	0	0
228	1	0
229	0	0
230	0	1

3.4 Validation

The classification report gives us a brief summary of our model.

```
              precision    recall  f1-score   support

           0       0.78      0.89      0.83       147
           1       0.75      0.57      0.65        84

    accuracy                           0.77       231
   macro avg       0.77      0.73      0.74       231
weighted avg       0.77      0.77      0.77       231
```

Here,

Precision: It is the score of accuracy that a label has been predicted correctly.

$$\text{Precision} = TP/(TP + FP)$$

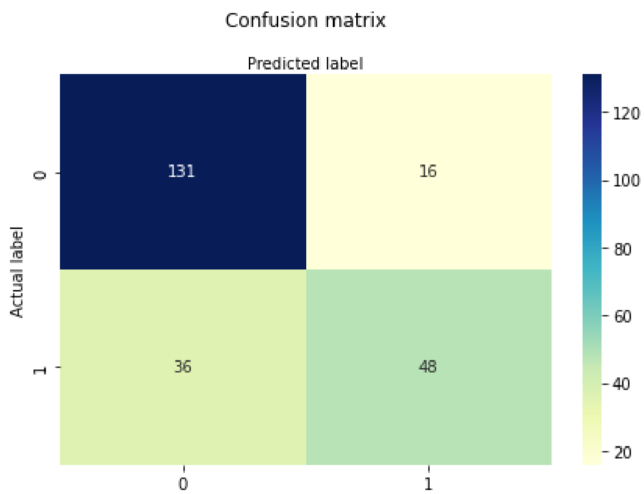
Recall: It is the true positive rate.

$$\text{Recall} = TP/(TP + FN)$$

F1 score: It is the harmonic mean between precision and recall. It has the best value at 1 and the worst at 0.

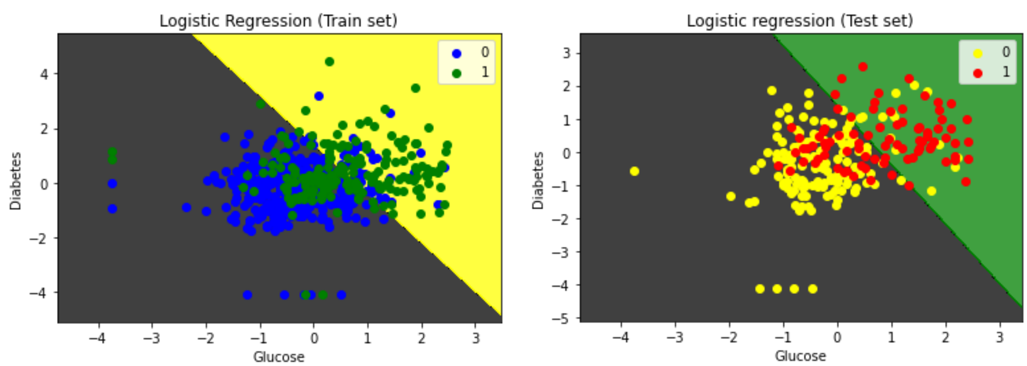
Weighted Average: It is the average accuracy of this model and is calculated as the average of the f1 score for both labels. It is 0.77 in our logistic regression model for the given data.

Using a confusion matrix to compare the results is a form of visualized comparison:

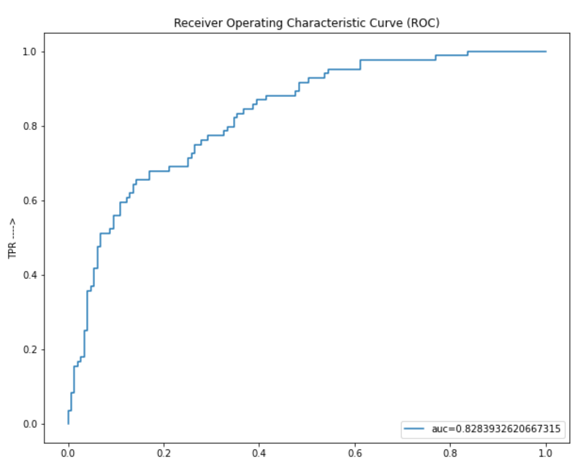


From the above figure, we can tell that there are 36 false negative and 16 false positive labels that have been marked incorrect.

We will create a scatter plot that will show us how the model performs:



ROC Curve plots specificicity to sensitivity of the model. More the area under the curve, better is our model. For our model the AUC is 0.82. In best case scenario it should be equal to 1.



References

[1] https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

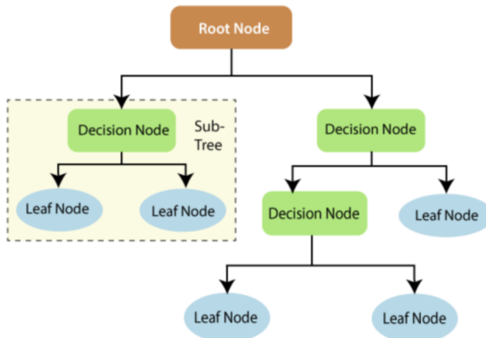
[2] https://www.educative.io/answers/what-is-sigmoid-and-its-role-in-logistic-regression

[3] https://medium.com/codex/machine-learning-logistic-regression-with-python-5ed4ded9d146

4 Decision Tree

A decision tree classifier is a supervised machine learning algorithm that is used for classification. It can with continuous as well as categorical data as well as a mix of both.

A decision tree splits the data on certain conditions into a binary tree with a root node at the top and branches in between and leaf nodes at the bottom[2].



A decision tree classifier splits the data into two on the concept of maximum information gain achieved by a feature. This is done iteratively from the root and at each level until pure children nodes are achieved for each branch.

Information gain is calculated as the measure of how much entropy is reduced by splitting the data on a given feature.

Entropy is the measure of randomness or impurity in the dataset.

$$\sum_{1=1}^{c} -p_i log_2 p_i$$

Thus the goal of information gain is to minimize entropy.

Gini index is also used to measure the impurity of node I, with n different classes of probability P.

$$G(p) = \sum_{k=1}^{K} p_k(1 - p_k) = 1 - \sum_{k=1}^{K} p_k^2$$

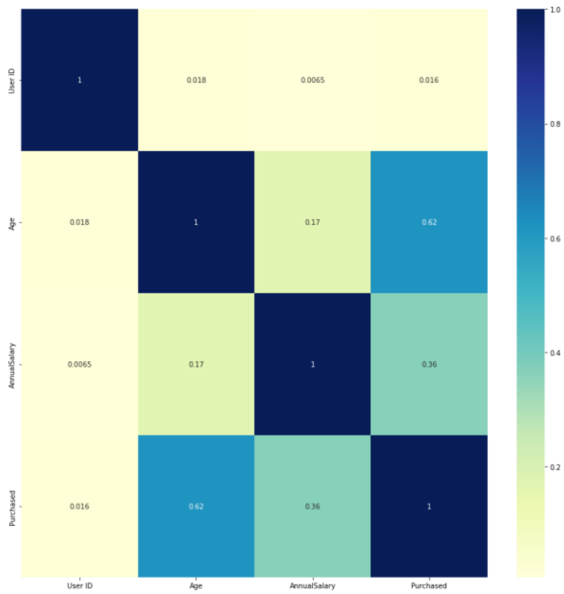
Among Gini index and Entropy, Gini Index is more efficient to be calculated.

4.1 Experiment

We are going to use a car sale dataset to build a decision tree classifier model which will predict if a customer will buy a car or not. The data has features as below:

```
RangeIndex: 1000 entries, 0 to 999
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   User ID       1000 non-null   int64
 1   Gender        1000 non-null   object
 2   Age           1000 non-null   int64
 3   AnnualSalary  1000 non-null   int64
 4   Purchased     1000 non-null   int64
dtypes: int64(4), object(1)
memory usage: 39.2+ KB
```

4.2 Feature Selection



We will create a correlation plot. Correlation helps us understand the relation between two variables. It ranges from -1 to +1. If the value of the relationship is zero, there is no correlation. As the value of correlation changes from zero to a positive or negative one, the linear relationship grows stronger. We can observe from the above plot that the best correlation for the outcome variable is from:

-Age

-AnnualSalary

As Age and AnnualSalary have a greater correlation with Purchased we choose these variables for our analysis. Additionally, we will also use the feature gender.

We will perform label encoding for the feature gender and set male as 1 and female as 0.

4.3 Training the model

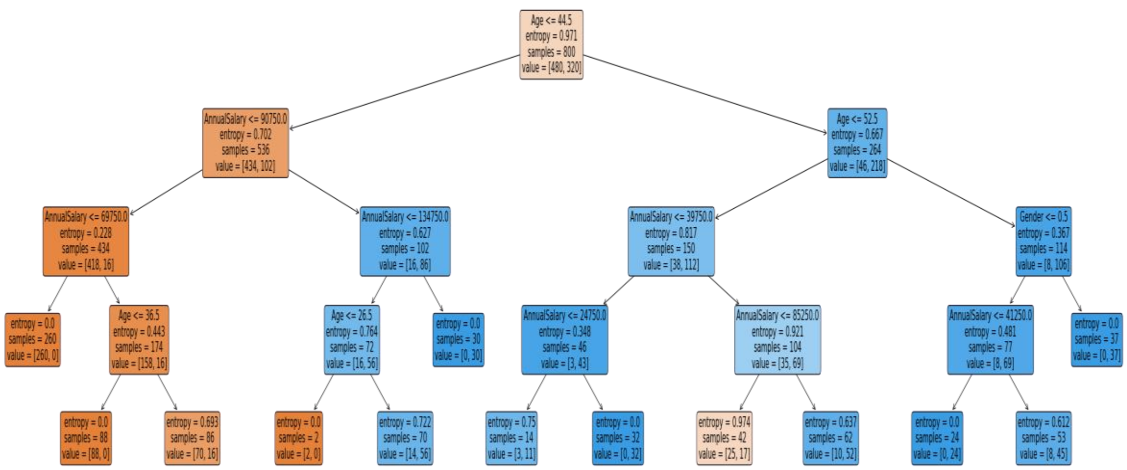
We will then store the Purchased variable(Y) and Age,AnnualSalary & Gender(X), and split this data into training and test in the ratio of 0.75 and .25 respectively. The train dataset is used for training our Decision tree classifier model.

Now we will train to fit our model on the train dataset and test the model to predict using the Test Dataset.

	Actual	Prediction
36	0	0
242	1	1
536	1	0
66	0	0
161	1	1
...	...	...
312	1	1
604	0	0
260	0	0
920	0	0
579	1	0

200 rows × 2 columns

4.4 Validation



The classification report gives us a brief summary of our model.

```
              precision    recall  f1-score   support

           0       0.88      0.97      0.92       118
           1       0.94      0.80      0.87        82

    accuracy                           0.90       200
   macro avg       0.91      0.89      0.89       200
weighted avg       0.90      0.90      0.90       200
```

Here,

Precision: It is the score of accuracy that a label has been predicted correctly.

$$\text{Precision} = TP/(TP + FP)$$

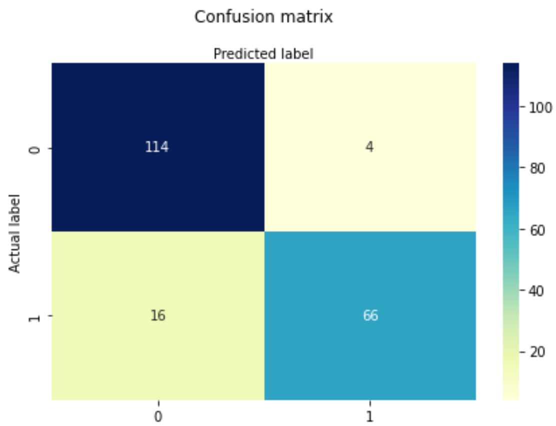
Recall: It is the true positive rate.

$$\text{Recall} = TP/(TP + FN)$$

F1 score: It is the harmonic mean between precision and recall. It has the best value at 1 and the worst at 0.

Weighted Average: It is the average accuracy of this model and is calculated as the average of the f1 score for both labels. It is 0.90 in our logistic regression model for the given data.

A classification report is not the best measure to find if the model is good or not. A better way is to use a confusion matrix:



From the above figure, we can tell that there are 16 false negative and 4 false positive labels that have been marked incorrect.

References

[1] https://www.kaggle.com/datasets/gabrielsantello/cars-purchase-decision-dataset

[2] https://medium.com/codex/decision-trees-in-python-98ca587f4329