Project Name: Heart Attack Risk Predictor

In this project we will Make an app which will help us predict the risk of a Heart Attack a person have.

We will do use various Algorithms to predict the result and see which one suits best and then we will use Auto ML Library EVAL ML to predict the results.

We will do the following things:

- Data Analysis
- Feature Engineering
- Satandardization
- Model Building
- Predictions

Let us import the necessary liabraries and read our DataSet

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

Let us import our Data Set

```
df= pd.read_csv("/content/heart.csv")
```

```
df= df.drop(['oldpeak','slp','thall'],axis=1)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 11 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
```

```
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trtbps    303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalachh  303 non-null    int64
 8   exng      303 non-null    int64
 9   caa       303 non-null    int64
 10  output    303 non-null    int64
dtypes: int64(11)
memory usage: 26.2 KB
```

```
df.head()
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	caa	output
0	63	1	3	145	233	1	0	150	0	0	1
1	37	1	2	130	250	0	1	187	0	0	1
2	41	0	1	130	204	0	0	172	0	0	1
3	56	1	1	120	236	0	1	178	0	0	1
4	57	0	0	120	354	0	1	163	1	0	1

Data Analysis

Understanding our DataSet:

Age : Age of the patient

Sex : Sex of the patient

exang: exercise induced angina (1 = yes; 0 = no)

ca: number of major vessels (0-3)

cp : Chest Pain type chest pain type

- Value 0: typical angina
- Value 1: atypical angina
- Value 2: non-anginal pain
- Value 3: asymptomatic

trtbps : resting blood pressure (in mm Hg) chol : cholestoral in mg/dl fetched via BMI sensor

fbs : (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)

rest_ecg : resting electrocardiographic results

- Value 0: normal
- Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)
- Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria

thalach : maximum heart rate achieved

target : 0= less chance of heart attack 1= more chance of heart attack

```
df.shape
```

```
(303, 11)
```

```
df.isnull().sum()
```

	0
age	0
sex	0
cp	0
trtbps	0
chol	0
fbs	0
restecg	0
thalachh	0
exng	0
caa	0
output	0

dtype: int64

As we can see there are no null values in our Data Set

```
df.corr()
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachl
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.39852
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	-0.058196	-0.04402
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444	0.044421	0.29576
trtbps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	-0.114103	-0.04669
chol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294	-0.151040	-0.00994
fbs	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.00856
restecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189	1.000000	0.04412
thalachh	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567	0.044123	1.00000
exng	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665	-0.070733	-0.37881
caa	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979	-0.072042	-0.21317
output	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	-0.028046	0.137230	0.42174

```
sns.heatmap(df.corr())
```

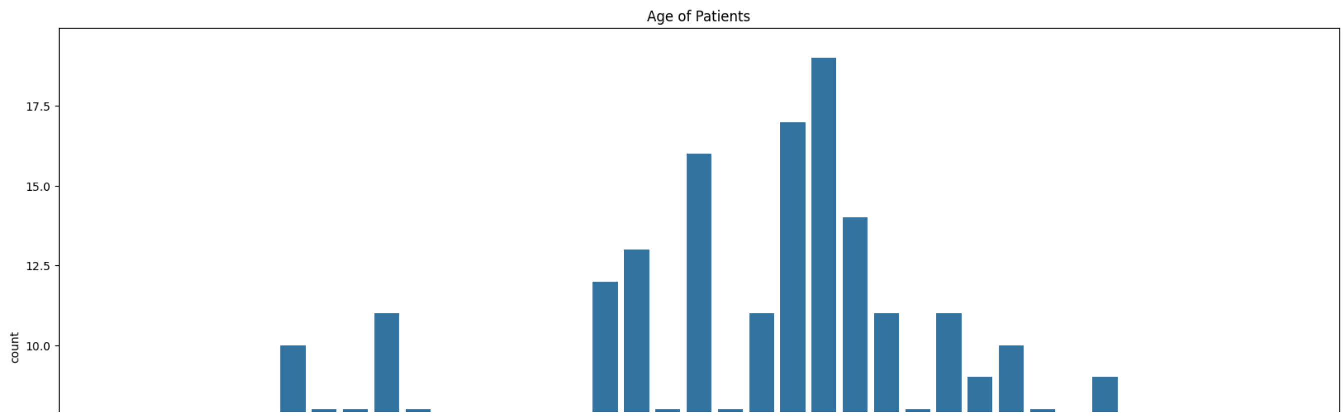
<Axes: >

As we can see our variables are not highly correlated to each other

We will do Uni and Bi variate analysis on our Features

```
plt.figure(figsize=(20, 10))
plt.title("Age of Patients")
plt.xlabel("Age")
sns.countplot(x='age',data=df)
```

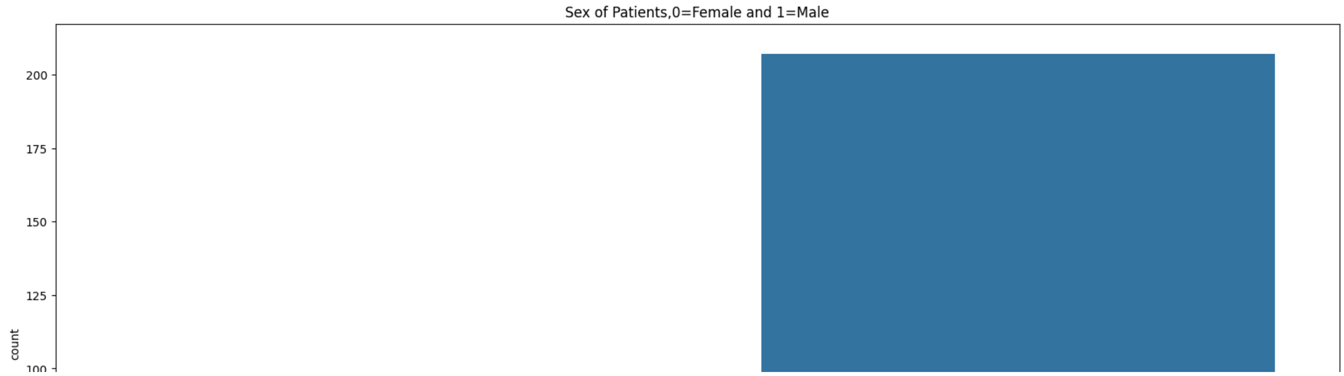
```
<Axes: title={'center': 'Age of Patients'}, xlabel='Age', ylabel='count'>
```



As we can see the Patients are of Age Group 51-67years in majority

```
plt.figure(figsize=(20, 10))
plt.title("Sex of Patients,0=Female and 1=Male")

sns.countplot(x='sex',data=df)
```

<Axes: title={'center': 'Sex of Patients,0=Female and 1=Male'}, xlabel='sex', ylabel='count'>

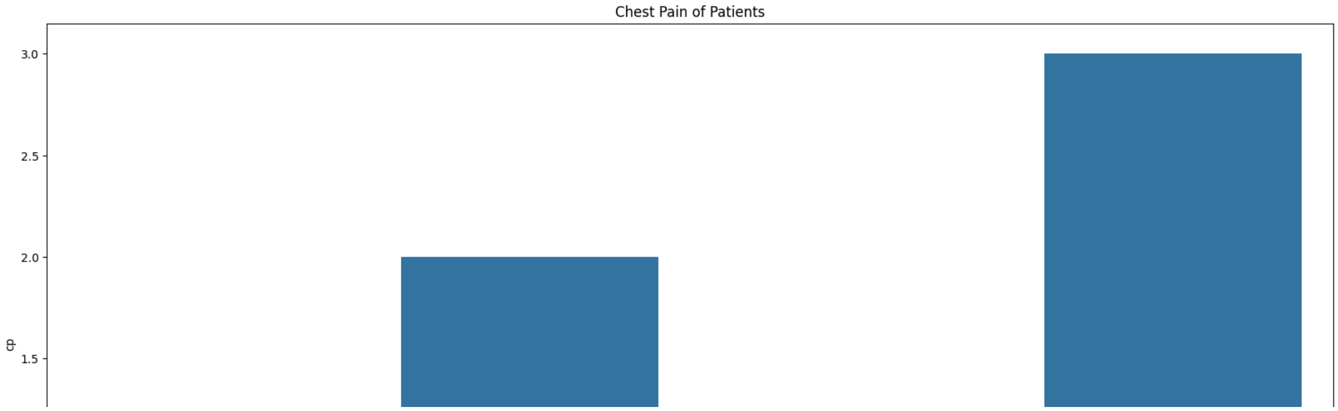



```
cp_data = df['cp'].value_counts().reset_index()
cp_data.loc[3, 'index'] = 'asymptomatic'
cp_data.loc[2, 'index'] = 'non-anginal'
cp_data.loc[1, 'index'] = 'Atyppical Anigma'
cp_data.loc[0, 'index'] = 'Typical Anigma'
cp_data
```

	cp	count	index
0	0	143	Typical Anigma
1	2	87	Atyppical Anigma
2	1	50	non-anginal
3	3	23	asymptomatic

```
plt.figure(figsize=(20, 10))
plt.title("Chest Pain of Patients")

sns.barplot(x=cp_data['index'],y= cp_data['cp'])
```

```
<Axes: title={'center': 'Chest Pain of Patients'}, xlabel='index', ylabel='cp'>
```



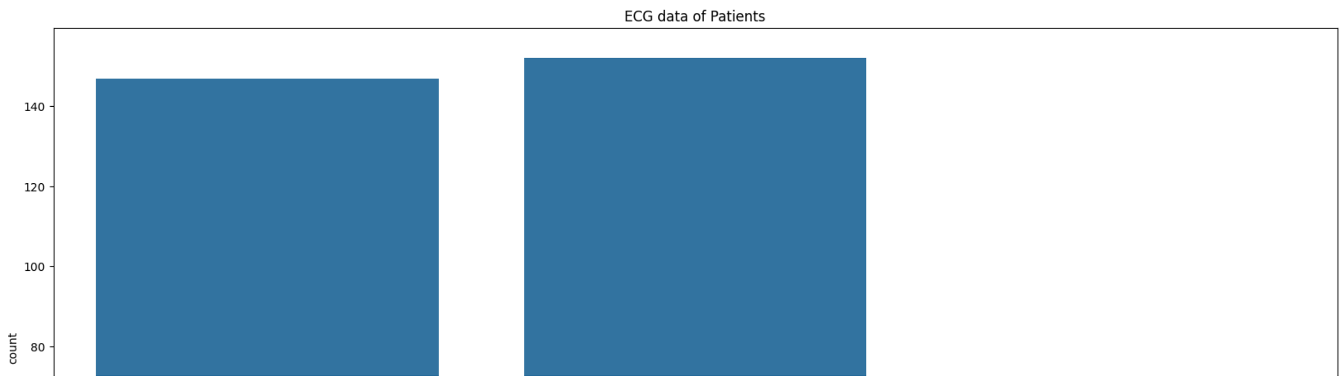
We have seen how the the Chest Pain Category is distributed

```
ecg_data= df['restecg'].value_counts().reset_index()
ecg_data[0,'index']= 'normal'
ecg_data[1,'index']= 'having ST-T wave abnormality'
ecg_data[2,'index']= 'showing probable or definite left ventricular hypertrophy by Estes'

ecg_data
```

	restecg	count	(0, index)	(1, index)	(2, index)
0	1	152	normal	having ST-T wave abnormality	showing probable or definite left ventricular ...
1	0	147	normal	having ST-T wave abnormality	showing probable or definite left ventricular ...

```
ecg_data = df['restecg'].value_counts().reset_index()
ecg_data.columns = ['restecg_type', 'count']

plt.figure(figsize=(20, 10))
plt.title("ECG data of Patients")
sns.barplot(x=ecg_data['restecg_type'], y=ecg_data['count'])
```

```
<Axes: title={'center': 'ECG data of Patients'}, xlabel='restecg_type',
ylabel='count'>
```



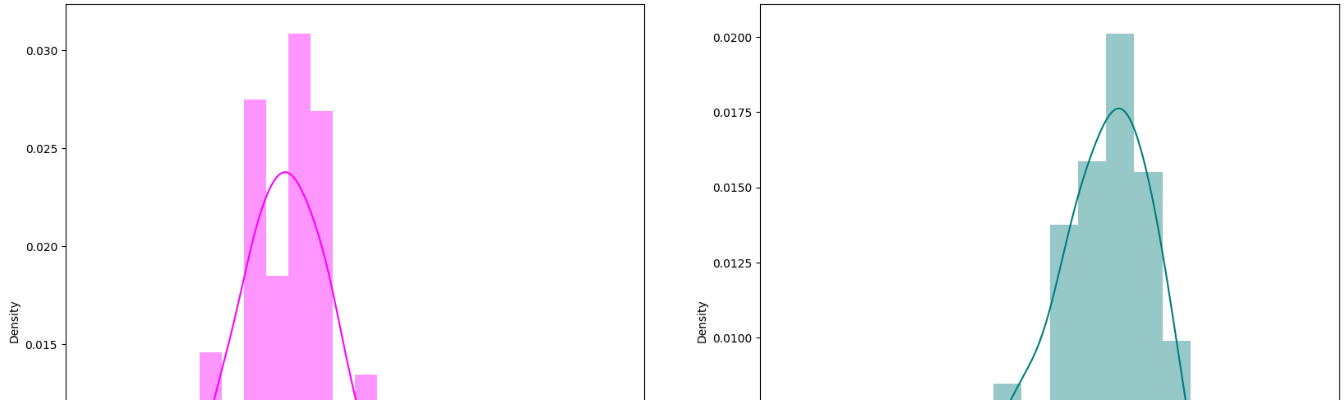

```
sns.pairplot(data=df, hue='output')
```

<seaborn.axisgrid.PairGrid at 0x78a437eae610>

Let us see for our Continuous Variable

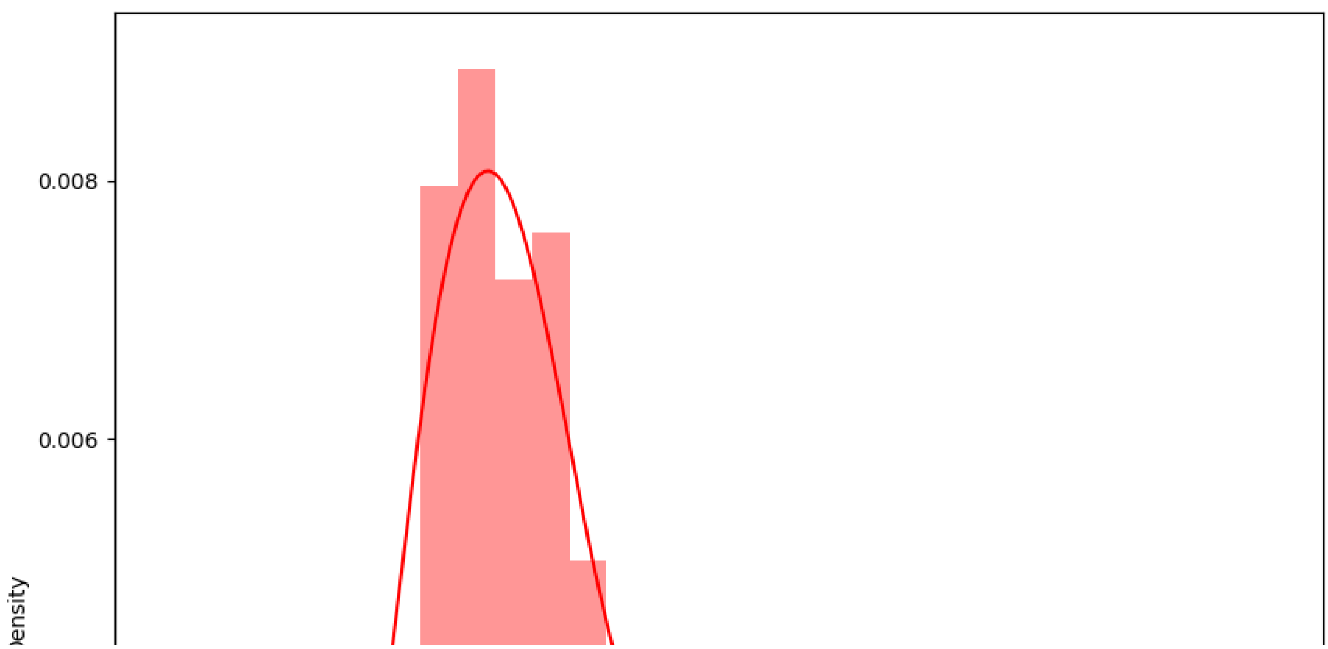
```
plt.figure(figsize=(20,10))
plt.subplot(1,2,1)
sns.distplot(df['trtbps'], kde=True, color = 'magenta')
plt.xlabel("Resting Blood Pressure (mmHg)")
plt.subplot(1,2,2)
sns.distplot(df['thalachh'], kde=True, color = 'teal')
plt.xlabel("Maximum Heart Rate Achieved (bpm)")
```

```
<ipython-input-35-17a8725cb836>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['trtbps'], kde=True, color = 'magenta')
<ipython-input-35-17a8725cb836>:6: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['thalachh'], kde=True, color = 'teal')
Text(0.5, 0, 'Maximum Heart Rate Achieved (bpm)')
```



```
plt.figure(figsize=(10,10))
sns.distplot(df['chol'], kde=True, color = 'red')
plt.xlabel("Cholestrol")
```

```
<ipython-input-36-ebe894739d0c>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['chol'], kde=True, color = 'red')
Text(0.5, 0, 'Cholestrol')
```



We have done the Analysis of the data now let's have a look at out data

```
df.head()
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	caa	output
0	63	1	3	145	233	1	0	150	0	0	1
1	37	1	2	130	250	0	1	187	0	0	1
2	41	0	1	130	204	0	0	172	0	0	1
3	56	1	1	120	236	0	1	178	0	0	1
4	57	0	0	120	354	0	1	163	1	0	1

Let us do Standardisation

```
from sklearn.preprocessing import StandardScaler
```

```
scale=StandardScaler()
```

```
scale.fit(df)
```

```
▾ StandardScaler
StandardScaler()
```

```
df= scale.transform(df)
```

```
df=pd.DataFrame(df,columns=['age', 'sex', 'cp', 'trtbps', 'chol', 'fbs', 'restecg', 'thalac
       'exng', 'caa', 'output'])
```

```
df.head()
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	
0	0.952197	0.681005	1.973123	0.763956	-0.256334	2.394438	-1.005832	0.015443	-0.69
1	-1.915313	0.681005	1.002577	-0.092738	0.072199	-0.417635	0.898962	1.633471	-0.69
2	-1.474158	-1.468418	0.032031	-0.092738	-0.816773	-0.417635	-1.005832	0.977514	-0.69
3	0.180175	0.681005	0.032031	-0.663867	-0.198357	-0.417635	0.898962	1.239897	-0.69
4	0.290464	-1.468418	-0.938515	-0.663867	2.082050	-0.417635	0.898962	0.583939	1.43

We can insert this data into our ML Models

We will use the following models for our predictions :

- Logistic Regression
- Decision Tree
- Random Forest
- K Nearest Neighbour
- SVM

Then we will use the ensembling techniques

Let us split our data

```
x= df.iloc[:,:-1]
x
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	
0	0.952197	0.681005	1.973123	0.763956	-0.256334	2.394438	-1.005832	0.015443	-0
1	-1.915313	0.681005	1.002577	-0.092738	0.072199	-0.417635	0.898962	1.633471	-0
2	-1.474158	-1.468418	0.032031	-0.092738	-0.816773	-0.417635	-1.005832	0.977514	-0
3	0.180175	0.681005	0.032031	-0.663867	-0.198357	-0.417635	0.898962	1.239897	-0
4	0.290464	-1.468418	-0.938515	-0.663867	2.082050	-0.417635	0.898962	0.583939	1
...	...	...	...	...	...	...	...	...	
298	0.290464	-1.468418	-0.938515	0.478391	-0.101730	-0.417635	0.898962	-1.165281	1
299	-1.033002	0.681005	1.973123	-1.234996	0.342756	-0.417635	0.898962	-0.771706	-0
300	1.503641	0.681005	-0.938515	0.706843	-1.029353	2.394438	0.898962	-0.378132	-0
301	0.290464	0.681005	-0.938515	-0.092738	-2.227533	-0.417635	0.898962	-1.515125	1
302	0.290464	-1.468418	0.032031	-0.092738	-0.198357	-0.417635	-1.005832	1.064975	-0

303 rows × 10 columns

```
y= df.iloc[:,-1:]
y
```

	output
0	0.914529
1	0.914529
2	0.914529
3	0.914529
4	0.914529
...	...
298	-1.093459
299	-1.093459
300	-1.093459
301	-1.093459
302	-1.093459

303 rows × 1 columns

```
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=101)
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.preprocessing import LabelEncoder
```

```
lbl= LabelEncoder()
```

```
encoded_y= lbl.fit_transform(y_train)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/preprocessing/_label.py:110: Dat
  y = column_or_1d(y, warn=True)
```

```
logreg= LogisticRegression()
```

```
logreg = LogisticRegression()
logreg.fit(x_train, encoded_y)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
encoded_y
```

```
array([0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0,
       1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0,
       0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1,
       1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1])
```

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
```

```
encoded_ytest= lbl.fit_transform(y_test)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/preprocessing/_label.py:110: Dat
  y = column_or_1d(y, warn=True)
```

```
Y_pred1 = logreg.predict(x_test)
lr_conf_matrix = confusion_matrix(encoded_ytest,Y_pred1 )
lr_acc_score = accuracy_score(encoded_ytest, Y_pred1)
```

```
lr_conf_matrix
```

```
array([[35,  9],
       [ 4, 43]])
```

```
print(lr_acc_score*100,"%")
```

```
85.71428571428571 %
```

As we see the Logistic Regression Model have a 85% accuracy

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
```

```
tree= DecisionTreeClassifier()
```

```
tree.fit(x_train,encoded_y)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
ypred2=tree.predict(x_test)
```

```
encoded_ytest= lbl.fit_transform(y_test)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/preprocessing/_label.py:110: Dat
  y = column_or_1d(y, warn=True)
```

```
tree_conf_matrix = confusion_matrix(encoded_ytest,ypred2 )
tree_acc_score = accuracy_score(encoded_ytest, ypred2)
```

```
tree_conf_matrix
```

```
array([[27, 17],
       [10, 37]])
```

```
print(tree_acc_score*100,"%")
```

```
70.32967032967034 %
```

As we see our Decision Tree Model does not perform well as it gives a score of only 69%

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
```

```
rf= RandomForestClassifier()
```

```
rf.fit(x_train,encoded_y)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
ypred3 = rf.predict(x_test)
```

```
rf_conf_matrix = confusion_matrix(encoded_ytest,ypred3 )
rf_acc_score = accuracy_score(encoded_ytest, ypred3)
```

```
rf_conf_matrix
```

```
array([[31, 13],
       [ 5, 42]])
```

```
print(rf_acc_score*100,"%")
```

```
80.21978021978022 %
```

RF also gives us an accuracy of around 80%

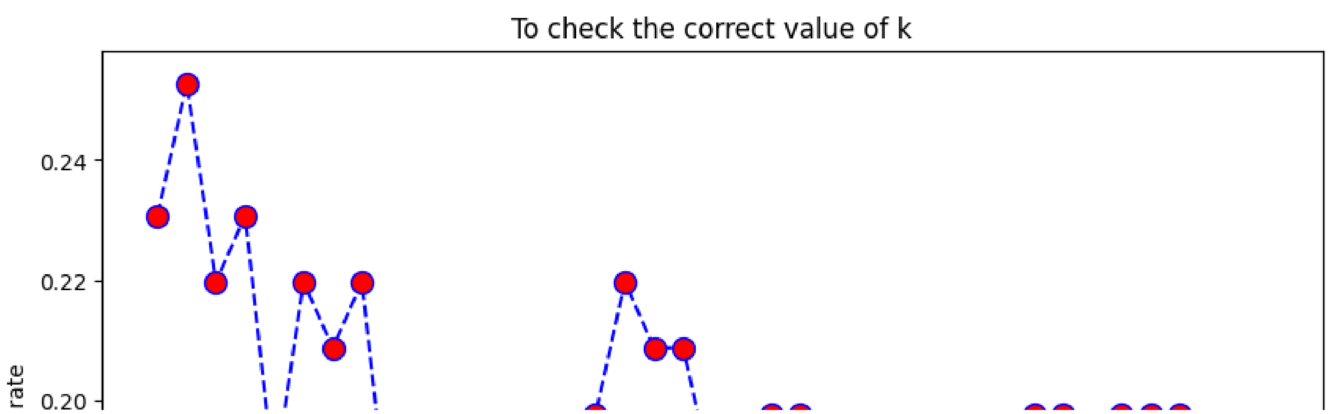
K Nearest Neighbour

We have to select what k we will use for the maximum accuracy

Let's write a function for it

```
from sklearn.neighbors import KNeighborsClassifier
```

```
error_rate= []
for i in range(1,40):
    knn= KNeighborsClassifier(n_neighbors=i)
    knn.fit(x_train,encoded_y)
    pred= knn.predict(x_test)
    error_rate.append(np.mean(pred != encoded_ytest))


plt.figure(figsize=(10,6))
plt.plot(range(1,40),error_rate,color='blue', linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=10)
plt.xlabel('K Vlaue')
plt.ylabel('Error rate')
plt.title('To check the correct value of k')
plt.show()
```



As we see from the graph we should select K= 12 as it gives the best error rate

```
knn= KNeighborsClassifier(n_neighbors=12)
knn.fit(x_train,encoded_y)
```

```
ypred4= knn.predict(x_test)
```

```
knn_conf_matrix = confusion_matrix(encoded_ytest,ypred4 )
knn_acc_score = accuracy_score(encoded_ytest, ypred4)
```

```
knn_conf_matrix
```

```
array([[35,  9],
       [ 5, 42]])
```

```
print(knn_acc_score*100,"%")
```

```
84.61538461538461 %
```

As we see KNN gives us an accuracy of around 85% which is good

Support Vector Machine(SVM)

```
from sklearn import svm
```

```
svm= svm.SVC()
```

```
svm.fit(x_train,encoded_y)
```

```
▾ SVC
SVC()
```

```
ypred5= svm.predict(x_test)
```

```
svm_conf_matrix = confusion_matrix(encoded_ytest,ypred5)
svm_acc_score = accuracy_score(encoded_ytest, ypred5)
```

```
svm_conf_matrix
```

```
array([[34, 10],
       [ 8, 39]])
```

```
print(svm_acc_score*100,"%")
```

```
80.21978021978022 %
```

We get an accuracy of 80% in SVM

Let us see our model accuracy in Table form

```
model_acc= pd.DataFrame({'Model' : ['Logistic Regression','Decision Tree','Random Forest','
```

```
model_acc = model_acc.sort_values(by=['Accuracy'],ascending=False)
```

```
model_acc
```

	Model	Accuracy
0	Logistic Regression	85.714286
3	K Nearest Neighbor	84.615385
2	Random Forest	80.219780
4	SVM	80.219780
1	Decision Tree	70.329670

Let us use one more Techniques known as Adaboost, this is a Boosting technique which uses multiple models for better accuracy.

Let us first use some random parameters for training the model without Hypertuning.

```
from sklearn.ensemble import AdaBoostClassifier
```

```
adab = AdaBoostClassifier(estimator=svm, n_estimators=100, algorithm='SAMME',
                          learning_rate=0.01, random_state=0)
```

```
adab.fit(x_train,encoded_y)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/ensemble/_weight_boosting.py:519
  warnings.warn(
```

▸ **AdaBoostClassifier** ⓘ ?

▸ **estimator: SVC**

▸ SVC ?

```
ypred6=adab.predict(x_test)
```

```
adab_conf_matrix = confusion_matrix(encoded_ytest,ypred6)
adab_acc_score = accuracy_score(encoded_ytest, ypred6)
```

```
adab_conf_matrix
```

```
array([[32, 12],
       [ 4, 43]])
```

```
print(adab_acc_score*100,"%")
```

```
82.41758241758241 %
```

```
adab.score(x_train,encoded_y)
```

```
0.8160377358490566
```

```
adab.score(x_test,encoded_ytest)
```

```
0.8241758241758241
```

As we see our model has performed very not bad with just 80% accuracy

We will use Grid Seach CV for HyperParameter Tuning

Grid Search CV

Let us try Grid Search CV for our top 3 performing Algorithms for HyperParameter tuning

```
from sklearn.model_selection import GridSearchCV
```

```
model_acc
```

	Model	Accuracy
0	Logistic Regression	85.714286
3	K Nearest Neighbor	84.615385
2	Random Forest	80.219780
4	SVM	80.219780
1	Decision Tree	70.329670

Logistic Regression

```
param_grid= {

    'solver': ['newton-cg', 'lbfgs', 'liblinear','sag', 'saga'],
    'penalty' : ['none', 'l1', 'l2', 'elasticnet'],
    'C' : [100, 10, 1.0, 0.1, 0.01]

}
```

```
grid1= GridSearchCV(LogisticRegression(),param_grid)
```

```
grid1.fit(x_train,encoded_y)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_validation.py:5
325 fits failed out of a total of 500.
The score on these train-test partitions for these parameters will be set to nan
If these failures are not expected, you can try to debug them by setting error_s

Below are more details about the failures:
--------------------------------------------------------------------------------
125 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_validat
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.11/dist-packages/sklearn/base.py", line 1382, in 
    estimator._validate_params()
  File "/usr/local/lib/python3.11/dist-packages/sklearn/base.py", line 436, in _
    validate_parameter_constraints(
  File "/usr/local/lib/python3.11/dist-packages/sklearn/utils/_param_validation.
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'penalty' parameter o

--------------------------------------------------------------------------------
25 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_validat
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.11/dist-packages/sklearn/base.py", line 1389, in 
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.p
    solver = _check_solver(self.solver, self.penalty, self.dual)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.p
    raise ValueError(
ValueError: Solver newton-cg supports only 'l2' or None penalties, got l1 penalt

--------------------------------------------------------------------------------
25 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_validat
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.11/dist-packages/sklearn/base.py", line 1389, in 
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.p
    solver = _check_solver(self.solver, self.penalty, self.dual)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.p
    raise ValueError(
ValueError: Solver lbfgs supports only 'l2' or None penalties, got l1 penalty.

--------------------------------------------------------------------------------
25 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_validat
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.11/dist-packages/sklearn/base.py", line 1389, in 
    return fit_method(estimator, *args, **kwargs)
```

```
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.p
    solver = _check_solver(self.solver, self.penalty, self.dual)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.p
    raise ValueError(
ValueError: Solver sag supports only 'l2' or None penalties, got l1 penalty.

--------------------------------------------------------------------------------
25 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_validat
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.11/dist-packages/sklearn/base.py", line 1389, in
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.p
    solver = _check_solver(self.solver, self.penalty, self.dual)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.p
    raise ValueError(
ValueError: Solver newton-cg supports only 'l2' or None penalties, got elasticne

--------------------------------------------------------------------------------
25 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_validat
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.11/dist-packages/sklearn/base.py", line 1389, in
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.p
    solver = _check_solver(self.solver, self.penalty, self.dual)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.p
    raise ValueError(
ValueError: Solver lbfgs supports only 'l2' or None penalties, got elasticnet pe

--------------------------------------------------------------------------------
25 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_validat
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.11/dist-packages/sklearn/base.py", line 1389, in
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.p
    solver = _check_solver(self.solver, self.penalty, self.dual)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.p
    raise ValueError(
ValueError: Only 'saga' solver supports elasticnet penalty, got solver=liblinear

--------------------------------------------------------------------------------
25 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_validat
```

```
grid1.best_params_
```

```
{'C': 0.01, 'penalty': 'l2', 'solver': 'liblinear'}
```

```
  return fit_method(estimator, *args, **kwargs)
         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.p
  solver = _check_solver(self.solver, self.penalty, self.dual)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.p
```

Let us apply these para in our Model

```
logreg1= LogisticRegression(C=0.01,penalty='l2',solver='liblinear')
```

```
logreg1.fit(x_train,encoded_y)
```

```
LogisticRegression
LogisticRegression(C=0.01, solver='liblinear')
```

```
/sklearn/model_selection/_validat
s)
/sklearn/base.py", line 1389, in
  return fit_method(estimator, *args, **kwargs)
```

```
logreg_pred= logreg1.predict(x_test)
```

```
  raise ValueError("l1_ratio must be specified when penalty is elasticnet.")
```

```
logreg_pred_conf_matrix = confusion_matrix(encoded_ytest,logreg_pred)
logreg_pred_acc_score = accuracy_score(encoded_ytest, logreg_pred)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_search.py:1108:
```

```
logreg_pred_conf_matrix
```

```
array([[33, 11],
       [ 6, 41]])
```

```
       nan        nan        nan        nan        nan        nan
       nan        nan        nan 0.78316722        nan 0.78316722
0.78316722 0.78316722 0.78316722 0.78316722 0.78316722        nan
       nan        nan        nan        nan        nan        nan
```

```
print(logreg_pred_acc_score*100,"%")
```

```
81.31868131868131 %
```

```
0.78316722        nan        nan        nan        nan        nan
       nan        nan        nan        nan        nan        nan
       nan 0.77364341        nan 0.76877076 0.78781838 0.78781838
0.78316722 0.78781838 0.78781838        nan        nan        nan
       nan        nan        nan        nan        nan        nan
       nan        nan        nan 0.44341085        nan 0.55658915
0.78316722 0.78316722 0.79756368 0.78316722 0.78316722        nan
       nan        nan        nan        nan]
  warnings.warn(
GridSearchCV
```

We got an accuracy of 81%

KNN

```
n_neighbors = range(1, 21, 2)
weights = ['uniform', 'distance']
metric = ['euclidean', 'manhattan', 'minkowski']
```

```
grid = dict(n_neighbors=n_neighbors,weights=weights,metric=metric)
```

```
from sklearn.model_selection import RepeatedStratifiedKFold
```

```
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
```

```
grid_search = GridSearchCV(estimator=knn, param_grid=grid, n_jobs=-1, cv=cv, scoring='accur
```

```
grid_search.fit(x_train,encoded_y)
```

```
GridSearchCV
best_estimator_:
KNeighborsClassifier
KNeighborsClassifier
```

```
grid_search.best_params_
```

```
{'metric': 'manhattan', 'n_neighbors': 11, 'weights': 'distance'}
```

Let's apply

```
knn= KNeighborsClassifier(n_neighbors=12,metric='manhattan',weights='distance')
knn.fit(x_train,encoded_y)
knn_pred= knn.predict(x_test)
```

```
knn_pred_conf_matrix = confusion_matrix(encoded_ytest,knn_pred)
knn_pred_acc_score = accuracy_score(encoded_ytest, knn_pred)
```

```
knn_pred_conf_matrix
```

```
array([[33, 11],
       [ 5, 42]])
```

```
print(knn_pred_acc_score*100,"%")
```

```
82.41758241758241 %
```

We have an Accuracy of 82.5%

SVM

```
kernel = ['poly', 'rbf', 'sigmoid']
C = [50, 10, 1.0, 0.1, 0.01]
gamma = ['scale']
```

```
grid = dict(kernel=kernel,C=C,gamma=gamma)
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
grid_search = GridSearchCV(estimator=svm, param_grid=grid, n_jobs=-1, cv=cv, scoring='accur
```

```
grid_search.fit(x_train,encoded_y)
```

GridSearchCV

best_estimator_:
SVC

SVC

```
grid_search.best_params_
```

```
{'C': 0.1, 'gamma': 'scale', 'kernel': 'sigmoid'}
```

Let us apply these

```
from sklearn.svm import SVC
```

```
svc= SVC(C= 0.1, gamma= 'scale',kernel= 'sigmoid')
```

```
svc.fit(x_train,encoded_y)
```

SVC

```
SVC(C=0.1, kernel='sigmoid')
```

```
svm_pred= svc.predict(x_test)
```

```
svm_pred_conf_matrix = confusion_matrix(encoded_ytest,svm_pred)
svm_pred_acc_score = accuracy_score(encoded_ytest, svm_pred)
```

```
svm_pred_conf_matrix
```

```
array([[32, 12],
       [ 5, 42]])
```

```
print(svm_pred_acc_score*100,"%")
```

```
81.31868131868131 %
```

Accuracy is 81%

Final Verdict

After comparing all the models the best performing model is :

Logistic Regression with no Hyperparameter tuning

```
logreg= LogisticRegression()
logreg = LogisticRegression()
logreg.fit(x_train, encoded_y)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
Y_pred1
```

```
array([0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1,
       0, 1, 0])
```

```
lr_conf_matrix
```

```
array([[35,  9],
       [ 4, 43]])
```

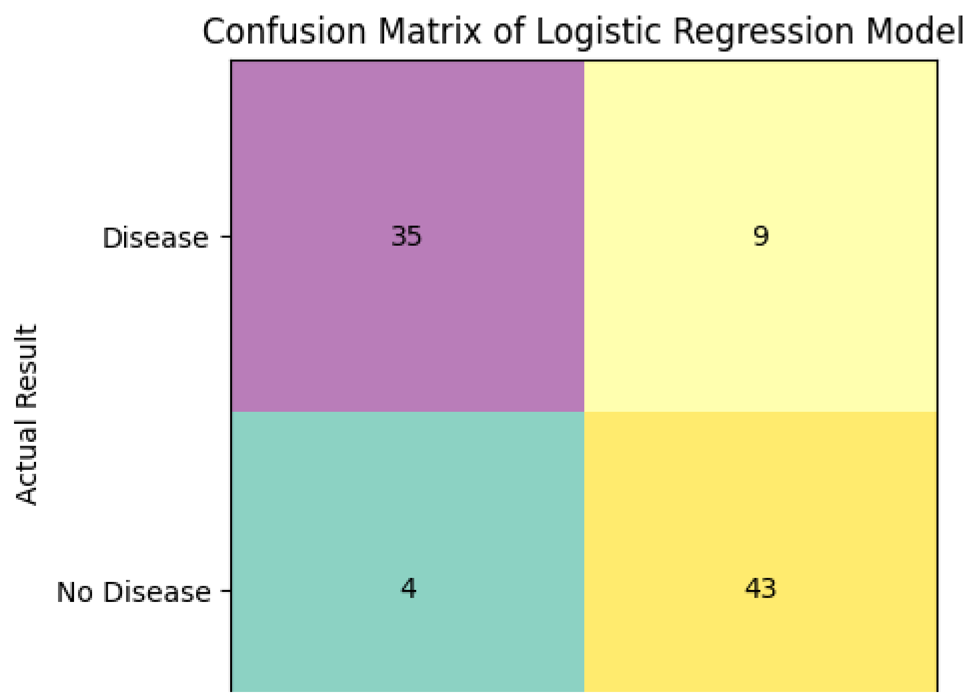
```
print(lr_acc_score*100,"%")
```

```
85.71428571428571 %
```

Let us build a proper confusion matrix for our model

```
# Confusion Matrix of  Model enlarged
options = ["Disease", 'No Disease']

fig, ax = plt.subplots()
im = ax.imshow(lr_conf_matrix, cmap= 'Set3', interpolation='nearest')

# We want to show all ticks...
ax.set_xticks(np.arange(len(options)))
ax.set_yticks(np.arange(len(options)))
# ... and label them with the respective list entries
```

```
ax.set_xticklabels(options)
ax.set_yticklabels(options)

# Rotate the tick labels and set their alignment.
plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
         rotation_mode="anchor")

# Loop over data dimensions and create text annotations.
for i in range(len(options)):
    for j in range(len(options)):
        text = ax.text(j, i, lr_conf_matrix[i, j],
                       ha="center", va="center", color="black")

ax.set_title("Confusion Matrix of Logistic Regression Model")
fig.tight_layout()
plt.xlabel('Model Prediction')
plt.ylabel('Actual Result')
plt.show()
print("ACCURACY of our model is ",lr_acc_score*100,"%")
```



We have succesfully made our model which predicts weather a person is having a risk of Heart Disease or not with 85.7% accuracy

```
import pickle
```

```
pickle.dump(logreg,open('heart.pkl','wb'))
```

Using Auto ML

EVAL ML :

EvalML is an open-source AutoML library written in python that automates a large part of the machine learning process and we can easily evaluate which machine learning pipeline works better for the given set of data.

Installing Eval ML

```
!pip install evalml
```

```
Requirement already satisfied: evalml in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: numpy>=1.22.0 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: pandas<2.1.0,>=1.5.0 in /usr/local/lib/python3.1
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: scikit-learn>=1.3.2 in /usr/local/lib/python3.11
Requirement already satisfied: scikit-optimize>=0.9.0 in /usr/local/lib/python3
Requirement already satisfied: pyzmq>=20.0.0 in /usr/local/lib/python3.11/dist-
```

```
Requirement already satisfied: colorama>=0.4.4 in /usr/local/lib/python3.11/dis
Requirement already satisfied: cloudpickle>=1.5.0 in /usr/local/lib/python3.11/
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: shap>=0.45.0 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: statsmodels>=0.12.2 in /usr/local/lib/python3.11
Requirement already satisfied: texttable>=1.6.2 in /usr/local/lib/python3.11/di
Requirement already satisfied: woodwork>=0.22.0 in /usr/local/lib/python3.11/di
Requirement already satisfied: dask!=2022.10.1,>=2022.2.0 in /usr/local/lib/pyt
Requirement already satisfied: distributed!=2022.10.1,>=2022.2.0 in /usr/local/
Requirement already satisfied: featuretools>=1.16.0 in /usr/local/lib/python3.1
Requirement already satisfied: nlp-primitives>=2.9.0 in /usr/local/lib/python3.
Requirement already satisfied: networkx>=2.7 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: plotly>=5.0.0 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: kaleido>=0.2.0 in /usr/local/lib/python3.11/dist
Requirement already satisfied: ipywidgets>=7.5 in /usr/local/lib/python3.11/dis
Requirement already satisfied: xgboost>=1.7.0.post0 in /usr/local/lib/python3.1
Requirement already satisfied: catboost>=1.1.1 in /usr/local/lib/python3.11/dis
Requirement already satisfied: lightgbm>=4.0.0 in /usr/local/lib/python3.11/dis
Requirement already satisfied: matplotlib>=3.3.3 in /usr/local/lib/python3.11/c
Requirement already satisfied: seaborn>=0.11.1 in /usr/local/lib/python3.11/dis
Requirement already satisfied: category-encoders<=2.5.1.post0,>=2.2.2 in /usr/l
Requirement already satisfied: imbalanced-learn>=0.11.0 in /usr/local/lib/pytho
Requirement already satisfied: pmdarima>=1.8.5 in /usr/local/lib/python3.11/dis
Requirement already satisfied: sktime<0.29.0,>=0.21.0 in /usr/local/lib/python3
Requirement already satisfied: lime>=0.2.0.1 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: tomli>=2.0.1 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: packaging>=23.0 in /usr/local/lib/python3.11/dis
Requirement already satisfied: black>=22.3.0 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: holidays>=0.13 in /usr/local/lib/python3.11/dist
Requirement already satisfied: graphviz>=0.13 in /usr/local/lib/python3.11/dist
Requirement already satisfied: mypy-extensions>=0.4.3 in /usr/local/lib/python3
Requirement already satisfied: pathspec>=0.9.0 in /usr/local/lib/python3.11/dis
Requirement already satisfied: platformdirs>=2 in /usr/local/lib/python3.11/dis
Requirement already satisfied: ipython>=7.8.0 in /usr/local/lib/python3.11/dist
Requirement already satisfied: tokenize-rt>=3.2.0 in /usr/local/lib/python3.11/
Requirement already satisfied: six in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: fsspec>=2021.09.0 in /usr/local/lib/python3.11/c
Requirement already satisfied: partd>=1.4.0 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: pyyaml>=5.3.1 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: toolz>=0.10.0 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: importlib_metadata>=4.13.0 in /usr/local/lib/pyt
Requirement already satisfied: jinja2>=2.10.3 in /usr/local/lib/python3.11/dist
Requirement already satisfied: locket>=1.0.0 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: msgpack>=1.0.2 in /usr/local/lib/python3.11/dist
Requirement already satisfied: psutil>=5.8.0 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: sortedcontainers>=2.0.5 in /usr/local/lib/pythor
Requirement already satisfied: tblib>=1.6.0 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: tornado>=6.2.0 in /usr/local/lib/python3.11/dist
Requirement already satisfied: urllib3>=1.26.5 in /usr/local/lib/python3.11/dis
```

Let us load our DataSet.

```
import pandas as pd
df= pd.read_csv("/content/heart.csv")
```

```
df.head()
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	

```
x= df.iloc[:,:-1]
x
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	th
0	63	1	3	145	233	1	0	150	0	2.3	0	0	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
298	57	0	0	140	241	0	1	123	1	0.2	1	0	
299	45	1	3	110	264	0	1	132	0	1.2	1	0	
300	68	1	0	144	193	1	1	141	0	3.4	1	2	
301	57	1	0	130	131	0	1	115	1	1.2	1	1	
302	57	0	1	130	236	0	0	174	0	0.0	1	1	

303 rows × 13 columns

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder

# Instantiate the LabelEncoder
lbl = LabelEncoder()

y= df.iloc[:,-1:]
y= lbl.fit_transform(y) # Use lbl instead of lb
y
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/preprocessing/_label.py:114: Dat
  y = column_or_1d(y, warn=True)
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

Importing Eval ML Library

```
import evalml
```

Eval ML Library will do all the pre processing techniques for us and split the data for us

```
X_train, X_test, y_train, y_test = evalml.preprocessing.split_data(x, y, problem_type='bina
```

There are different problem type parameters in Eval ML, we have a Binary type problem here, that's why we are using Binary as a input

Running the Auto ML to select best Algorithm

```
from evalml.automl import AutoMLSearch
automl = AutoMLSearch(X_train=X_train, y_train=y_train, problem_type='binary')
automl.search()
```

```
{1: {'Random Forest Classifier w/ Label Encoder + Imputer + RF Classifier Select
From Model': 2.6637513637542725,
  'Total time of batch': 2.7914462089538574},
 2: {'LightGBM Classifier w/ Label Encoder + Imputer + Select Columns
Transformer': 1.298210859298706,
  'Extra Trees Classifier w/ Label Encoder + Imputer + Select Columns
Transformer': 1.9191703796386719,
  'Elastic Net Classifier w/ Label Encoder + Imputer + Standard Scaler + Select
Columns Transformer': 2.190389394760132,
  'XGBoost Classifier w/ Label Encoder + Imputer + Select Columns Transformer':
2.1611223220825195,
  'Logistic Regression Classifier w/ Label Encoder + Imputer + Standard Scaler +
```

```
Select Columns Transformer': 3.439563274383545,
  'Total time of batch': 11.627403020858765}}
```

As we see from the above output thge Auto ML Classifier has given us the best fit Algorithm which is Extra Trees Classifier with Imputer We can also commpare the rest of the models

```
automl.rankings
```

	id	pipeline_name	search_order	ranking_score	mean_cv_score	standard_deviat:
0	3	Extra Trees Classifier w/ Label Encoder + Impu...	3	0.413358	0.413358	
1	2	LightGBM Classifier w/ Label Encoder + Imputer...	2	0.462099	0.462099	
2	1	Random Forest Classifier w/ Label Encoder + Im...	1	0.466918	0.466918	
3	6	Logistic Regression Classifier w/ Label Encode...	6	0.469250	0.469250	
4	4	Elastic Net Classifier w/ Label Encoder + Impu...	4	0.470037	0.470037	
5	5	XGBoost Classifier w/ Label Encoder + Imputer ...	5	0.499149	0.499149	
6	0	Mode Baseline Binary Classification Pipeline	0	16.382805	16.382805	

```
automl.best_pipeline
```

```
pipeline = BinaryClassificationPipeline(component_graph={'Label Encoder':
['Label Encoder', 'X', 'y'], 'Imputer': ['Imputer', 'X', 'Label Encoder.y'],
'Select Columns Transformer': ['Select Columns Transformer', 'Imputer.x', 'Label
Encoder.y'], 'Extra Trees Classifier': ['Extra Trees Classifier', 'Select
Columns Transformer.x', 'Label Encoder.y']}, parameters={'Label Encoder':
```

```
{'positive_label': None}, 'Imputer':{'categorical_impute_strategy':
'most_frequent', 'numeric_impute_strategy': 'mean', 'boolean_impute_strategy':
'most_frequent', 'categorical_fill_value': None, 'numeric_fill_value': None,
'boolean_fill_value': None}, 'Select Columns Transformer':{'columns': ['age',
'cp', 'thalachh', 'exng', 'oldpeak', 'caa', 'thall']}, 'Extra Trees Classifier':
{'n_estimators': 100, 'max_features': 'sqrt', 'max_depth': 6,
'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_jobs': -1}},
random_seed=0)
```

```
best_pipeline=automl.best_pipeline
```

We can have a Detailed description of our Best Selected Model

```
automl.describe_pipeline(automl.rankings.iloc[0]["id"])
```

```
*******************************************************************************
INFO:evalml.pipelines.pipeline_base.describe:
*******************************************************************************
* Extra Trees Classifier w/ Label Encoder + Imputer + Select Columns Transforme
INFO:evalml.pipelines.pipeline_base.describe:* Extra Trees Classifier w/ Label
*******************************************************************************
INFO:evalml.pipelines.pipeline_base.describe:**********************************

INFO:evalml.pipelines.pipeline_base.describe:
Problem Type: binary
INFO:evalml.pipelines.pipeline_base.describe:Problem Type: binary
Model Family: Extra Trees
INFO:evalml.pipelines.pipeline_base.describe:Model Family: Extra Trees

INFO:evalml.pipelines.pipeline_base.describe:
Pipeline Steps
INFO:evalml.pipelines.pipeline_base.describe:Pipeline Steps
==============
INFO:evalml.pipelines.pipeline_base.describe:==============
1. Label Encoder
INFO:evalml.pipelines.component_graph.describe:1. Label Encoder
         * positive_label : None
INFO:evalml.pipelines.components.component_base.describe:        * positive_lab
2. Imputer
INFO:evalml.pipelines.component_graph.describe:2. Imputer
         * categorical_impute_strategy : most_frequent
INFO:evalml.pipelines.components.component_base.describe:        * categorical_
         * numeric_impute_strategy : mean
INFO:evalml.pipelines.components.component_base.describe:        * numeric_impu
         * boolean_impute_strategy : most_frequent
INFO:evalml.pipelines.components.component_base.describe:        * boolean_impu
         * categorical_fill_value : None
INFO:evalml.pipelines.components.component_base.describe:        * categorical_
         * numeric_fill_value : None
INFO:evalml.pipelines.components.component_base.describe:        * numeric_fill
         * boolean_fill_value : None
INFO:evalml.pipelines.components.component_base.describe:        * boolean_fill
```

```
3. Select Columns Transformer
INFO:evalml.pipelines.component_graph.describe:3. Select Columns Transformer
         * columns : ['age', 'cp', 'thalachh', 'exng', 'oldpeak', 'caa', 'thall
INFO:evalml.pipelines.components.component_base.describe:        * columns : ['
4. Extra Trees Classifier
INFO:evalml.pipelines.component_graph.describe:4. Extra Trees Classifier
         * n_estimators : 100
INFO:evalml.pipelines.components.component_base.describe:        * n_estimators
         * max_features : sqrt
INFO:evalml.pipelines.components.component_base.describe:        * max_features
         * max_depth : 6
INFO:evalml.pipelines.components.component_base.describe:        * max_depth :
         * min_samples_split : 2
INFO:evalml.pipelines.components.component_base.describe:        * min_samples_
         * min_weight_fraction_leaf : 0.0
INFO:evalml.pipelines.components.component_base.describe:        * min_weight_f
         * n_jobs : -1
INFO:evalml.pipelines.components.component_base.describe:        * n_jobs : -1
```

```
best_pipeline.score(X_test, y_test, objectives=["auc","f1","Precision","Recall"])
```

```
OrderedDict([('AUC', 0.8701298701298702),
             ('F1', 0.78125),
             ('Precision', 0.8064516129032258),
             ('Recall', 0.7575757575757576)])
```

Now if we want to build our Model for a specific objective we can do that

```
automl_auc = AutoMLSearch(X_train=X_train, y_train=y_train,
                          problem_type='binary',
                          objective='auc',
                          additional_objectives=['f1', 'precision'],
                          max_batches=1,
                          optimize_thresholds=True)

automl_auc.search()
```

```
{1: {'Random Forest Classifier w/ Label Encoder + Imputer + RF Classifier Select
From Model': 4.003447532653809,
  'Total time of batch': 4.1510655879974365}}
```

```
automl_auc.rankings
```

	id	pipeline_name	search_order	ranking_score	mean_cv_score	standard_deviat
		Random Forest Classifier w/				
1	0	Mode Baseline Binary Classification Pipeline	0	0.500000	0.500000	

```
automl_auc.describe_pipeline(automl_auc.rankings.iloc[0]["id"])
```

```
*****************************************************************************
INFO:evalml.pipelines.pipeline_base.describe:
*****************************************************************************
* Random Forest Classifier w/ Label Encoder + Imputer + RF Classifier Select Fr
INFO:evalml.pipelines.pipeline_base.describe:* Random Forest Classifier w/ Labe
*****************************************************************************
INFO:evalml.pipelines.pipeline_base.describe:**********************************

INFO:evalml.pipelines.pipeline_base.describe:
Problem Type: binary
INFO:evalml.pipelines.pipeline_base.describe:Problem Type: binary
Model Family: Random Forest
INFO:evalml.pipelines.pipeline_base.describe:Model Family: Random Forest

INFO:evalml.pipelines.pipeline_base.describe:
Pipeline Steps
INFO:evalml.pipelines.pipeline_base.describe:Pipeline Steps
==============
INFO:evalml.pipelines.pipeline_base.describe:==============
1. Label Encoder
INFO:evalml.pipelines.component_graph.describe:1. Label Encoder
         * positive_label : None
INFO:evalml.pipelines.components.component_base.describe:        * positive_lab
2. Imputer
INFO:evalml.pipelines.component_graph.describe:2. Imputer
         * categorical_impute_strategy : most_frequent
INFO:evalml.pipelines.components.component_base.describe:        * categorical_
         * numeric_impute_strategy : mean
INFO:evalml.pipelines.components.component_base.describe:        * numeric_impu
         * boolean_impute_strategy : most_frequent
INFO:evalml.pipelines.components.component_base.describe:        * boolean_impu
         * categorical_fill_value : None
INFO:evalml.pipelines.components.component_base.describe:        * categorical_
         * numeric_fill_value : None
```