Lab 10 : Model Selction

Import Commomn Libraries

In [1]:
```
import itertools
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import NullFormatter
import pandas as pd
import numpy as np
import matplotlib.ticker as ticker
from sklearn import preprocessing
%matplotlib inline
```

About dataset

This dataset is about past customer orders. The **customer_orders.csv** data set includes details of 1000 orders that are either delivered or not delivered. It includes following fields:

Field	Description
order_status	Whether the order was delivered or not delivered
order_amount	Total amount of the order
delivery_terms_days	Number of days allocated for delivery (e.g., 7, 15, or 30 days)
order_date	The date when the order was placed
delivery_date	The expected delivery date based on the delivery terms
customer_age	Age of the customer who placed the order
membership_level	Customer's membership tier (Bronze, Silver, or Gold)
customer_gender	Gender of the customer (Male or Female)

Import Training Dataset

In [2]:
```
df = pd.read_csv('customer_orders.csv')
df.head(20)
```

Out[2]:

	order_status	order_amount	delivery_terms_days	order_date	delivery_date	customer_age	membership_level	customer_g
0	NOT_DELIVERED	402	7	2021-07-06	2021-07-13	36	Gold	
1	DELIVERED	735	7	2021-04-12	2021-04-19	30	Silver	F
2	DELIVERED	570	7	2021-02-28	2021-03-07	37	Gold	
3	DELIVERED	406	30	2021-06-05	2021-07-05	29	Bronze	
4	NOT_DELIVERED	371	15	2021-02-17	2021-03-04	37	Gold	
5	DELIVERED	1000	15	2021-09-12	2021-09-27	44	Bronze	F
6	DELIVERED	320	30	2021-06-16	2021-07-16	31	Gold	
7	NOT_DELIVERED	914	15	2021-07-13	2021-07-28	36	Bronze	
8	DELIVERED	421	30	2021-02-23	2021-03-25	45	Bronze	F
9	DELIVERED	766	7	2021-02-20	2021-02-27	28	Bronze	F
10	DELIVERED	514	30	2021-08-01	2021-08-31	29	Bronze	
11	DELIVERED	688	30	2021-12-13	2022-01-12	60	Silver	
12	DELIVERED	437	15	2021-07-13	2021-07-28	18	Gold	F
13	DELIVERED	742	7	2021-05-21	2021-05-28	43	Bronze	F
14	DELIVERED	403	15	2021-08-30	2021-09-14	39	Bronze	F
15	DELIVERED	616	7	2021-08-11	2021-08-18	52	Silver	F
16	DELIVERED	948	7	2021-07-30	2021-08-06	24	Gold	F
17	DELIVERED	825	7	2021-10-19	2021-10-26	27	Gold	
18	DELIVERED	676	7	2021-05-21	2021-05-28	35	Bronze	F
19	DELIVERED	452	7	2021-05-12	2021-05-19	52	Gold	

EDA Steps

In [3]:
```
df.shape
```

Out[3]:
```
(1000, 8)
```

In [4]:
```
df.columns
```

Out[4]:
```
Index(['order_status', 'order_amount', 'delivery_terms_days', 'order_date',
       'delivery_date', 'customer_age', 'membership_level', 'customer_gender'],
      dtype='object')
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   order_status         1000 non-null   object
 1   order_amount         1000 non-null   int64 
 2   delivery_terms_days  1000 non-null   int64 
 3   order_date           1000 non-null   object
 4   delivery_date        1000 non-null   object
 5   customer_age         1000 non-null   int64 
 6   membership_level     1000 non-null   object
 7   customer_gender      1000 non-null   object
dtypes: int64(3), object(5)
memory usage: 62.6+ KB
```

In [6]:
```
df.describe()
```

Out[6]:

	order_amount	delivery_terms_days	customer_age
count	1000.000000	1000.000000	1000.000000
mean	642.994000	17.199000	38.232000
std	205.765754	9.548223	12.356977
min	301.000000	7.000000	18.000000
25%	454.500000	7.000000	28.000000
50%	633.000000	15.000000	37.000000
75%	822.000000	30.000000	49.000000
max	1000.000000	30.000000	60.000000

In [7]:
```
df.describe(include = 'object')
```

Out[7]:

	order_status	order_date	delivery_date	membership_level	customer_gender
count	1000	1000	1000	1000	1000
unique	2	341	348	3	2
top	DELIVERED	2021-09-20	2021-10-20	Bronze	Male
freq	900	8	11	362	501

Convert to date time object

In [8]:
```
df['order_date'] = pd.to_datetime(df['order_date'])
df['delivery_date'] = pd.to_datetime(df['delivery_date'])
df.head()
```

Out[8]:

	order_status	order_amount	delivery_terms_days	order_date	delivery_date	customer_age	membership_level	customer_ge
0	NOT_DELIVERED	402	7	2021-07-06	2021-07-13	36	Gold	
1	DELIVERED	735	7	2021-04-12	2021-04-19	30	Silver	Fe
2	DELIVERED	570	7	2021-02-28	2021-03-07	37	Gold	
3	DELIVERED	406	30	2021-06-05	2021-07-05	29	Bronze	
4	NOT_DELIVERED	371	15	2021-02-17	2021-03-04	37	Gold	

Data visualization and pre-processing Steps

Let's see how many of each class is in our data set

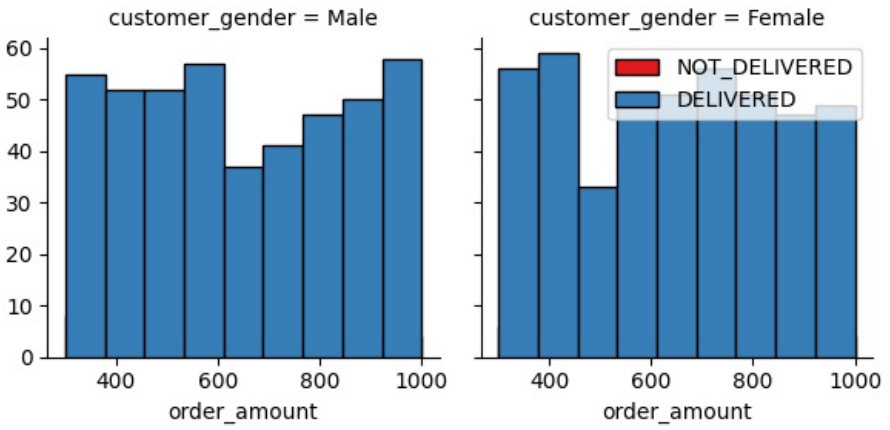
In [9]:
```
df['order_status'].value_counts()
```

Out[9]:
```
order_status
DELIVERED        900
NOT_DELIVERED    100
Name: count, dtype: int64
```

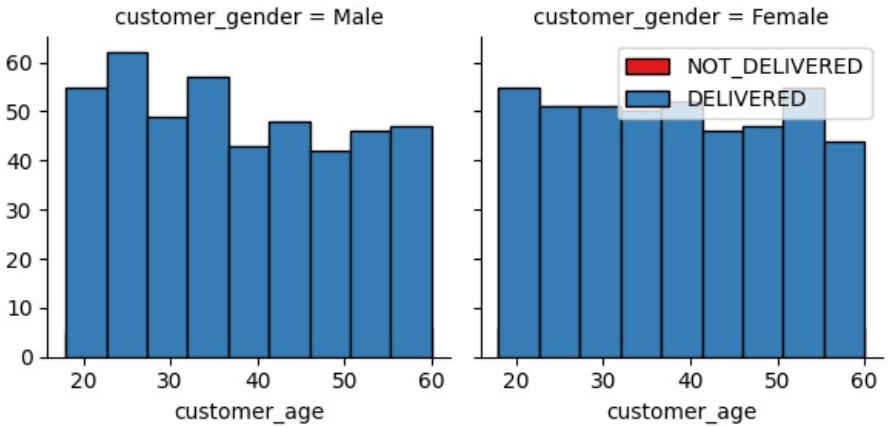
7970 items delivered while 330 items are not delivered

Let's plot some columns to underestand data better:

In [10]:
```
import seaborn as sns

bins = np.linspace(df.order_amount.min(),
                   df.order_amount.max(), 10)
g = sns.FacetGrid(df, col="customer_gender",
                  hue="order_status",
                  palette="Set1",
                  col_wrap=2)
g.map(plt.hist,
      'order_amount',
      bins=bins,
      ec="k")

g.axes[-1].legend()
plt.show()
```



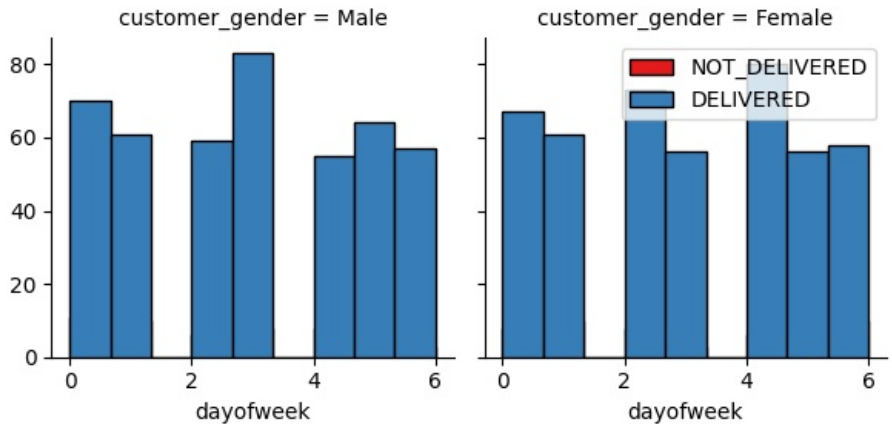
In [11]:
```
bins = np.linspace(df.customer_age.min(),
                   df.customer_age.max(), 10)
g = sns.FacetGrid(df,
                  col="customer_gender",
                  hue="order_status",
                  palette="Set1",
                  col_wrap=2)
g.map(plt.hist,
      'customer_age',
      bins=bins,
      ec="k")

g.axes[-1].legend()
plt.show()
```



Pre-processing: Feature selection/extraction

Let's look at the day of the week people order product

In [12]:
```
df['dayofweek'] = df['order_date'].dt.dayofweek
bins = np.linspace(df.dayofweek.min(),
                   df.dayofweek.max(),
                   10)
g = sns.FacetGrid(df,
                  col="customer_gender",
                  hue="order_status",
                  palette="Set1",
                  col_wrap=2)
g.map(plt.hist,
      'dayofweek',
      bins=bins,
      ec="k")
g.axes[-1].legend()
plt.show()
```



We see that people who get the loan at the end of the week don't pay it off, so let's use Feature binarization to set a threshold value less than day 4

In [13]:
```
df['weekend'] = df['dayofweek'].apply(lambda x: 1 if (x>3)  else 0)
df.head()
```

Out[13]:

	order_status	order_amount	delivery_terms_days	order_date	delivery_date	customer_age	membership_level	customer_ge
0	NOT_DELIVERED	402	7	2021-07-06	2021-07-13	36	Gold	
1	DELIVERED	735	7	2021-04-12	2021-04-19	30	Silver	Fe
2	DELIVERED	570	7	2021-02-28	2021-03-07	37	Gold	
3	DELIVERED	406	30	2021-06-05	2021-07-05	29	Bronze	
4	NOT_DELIVERED	371	15	2021-02-17	2021-03-04	37	Gold	

Convert Categorical features to numerical values

Let's look at gender:

In [14]:
```
df.groupby(['customer_gender'])['order_status'].value_counts(normalize=True)
```

Out[14]:
```
customer_gender  order_status
Female           DELIVERED        0.903808
                 NOT_DELIVERED    0.096192
Male             DELIVERED        0.896208
                 NOT_DELIVERED    0.103792
Name: proportion, dtype: float64
```

86 % of female pay there loans while only 73 % of males pay there loan

Let's convert male to 0 and female to 1:

In [15]:
```
df['customer_gender'].replace(to_replace=['Male','Female'], value=[0,1],inplace=True)
df.head()
```

```
C:\Users\Mohammed Meraj\AppData\Local\Temp\ipykernel_9696\4071422569.py:1: FutureWarning: A value is trying to b
e set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on w
hich we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)'
or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  df['customer_gender'].replace(to_replace=['Male','Female'], value=[0,1],inplace=True)
C:\Users\Mohammed Meraj\AppData\Local\Temp\ipykernel_9696\4071422569.py:1: FutureWarning: Downcasting behavior i
n `replace` is deprecated and will be removed in a future version. To retain the old behavior, explicitly call `
result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('future.no_silent_downca
sting', True)`
  df['customer_gender'].replace(to_replace=['Male','Female'], value=[0,1],inplace=True)
```

Out[15]:

	order_status	order_amount	delivery_terms_days	order_date	delivery_date	customer_age	membership_level	customer_ge
0	NOT_DELIVERED	402	7	2021-07-06	2021-07-13	36	Gold	
1	DELIVERED	735	7	2021-04-12	2021-04-19	30	Silver	
2	DELIVERED	570	7	2021-02-28	2021-03-07	37	Gold	
3	DELIVERED	406	30	2021-06-05	2021-07-05	29	Bronze	
4	NOT_DELIVERED	371	15	2021-02-17	2021-03-04	37	Gold	

One Hot Encoding

How about education?

In [16]:
```
df.groupby(['membership_level'])['order_status'].value_counts(normalize=True)
```

Out[16]:
```
membership_level  order_status
Bronze            DELIVERED        0.911602
                  NOT_DELIVERED    0.088398
Gold              DELIVERED        0.900302
                  NOT_DELIVERED    0.099698
Silver            DELIVERED        0.885993
                  NOT_DELIVERED    0.114007
Name: proportion, dtype: float64
```

Features before One Hot Encoding

In [17]:
```
df[['order_amount','delivery_terms_days','customer_age','customer_gender','membership_level']].head()
```

Out[17]:

	order_amount	delivery_terms_days	customer_age	customer_gender	membership_level
0	402	7	36	0	Gold
1	735	7	30	1	Silver
2	570	7	37	0	Gold
3	406	30	29	0	Bronze
4	371	15	37	0	Gold

Use one hot encoding technique to conver categorical varables to binary variables and append them to the feature Data Frame

In [18]:
```
Feature = df[['order_amount', 'delivery_terms_days', 'customer_age', 'customer_gender', 'weekend']]
Feature = pd.concat([Feature, pd.get_dummies(df['membership_level'])], axis=1)
print(Feature.head())
print(Feature.info())
```

```
   order_amount  delivery_terms_days  customer_age  customer_gender  weekend  \
0           402                    7            36                0        0
1           735                    7            30                1        0
2           570                    7            37                0        1
3           406                   30            29                0        1
4           371                   15            37                0        0

   Bronze   Gold  Silver
0   False   True   False
1   False  False    True
2   False   True   False
3    True  False   False
4   False   True   False
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   order_amount         1000 non-null   int64
 1   delivery_terms_days  1000 non-null   int64
 2   customer_age         1000 non-null   int64
 3   customer_gender      1000 non-null   int64
 4   weekend              1000 non-null   int64
 5   Bronze               1000 non-null   bool
 6   Gold                 1000 non-null   bool
 7   Silver               1000 non-null   bool
dtypes: bool(3), int64(5)
memory usage: 42.1 KB
None
```

Feature Selection

Let's define feature sets, X:

In [19]:
```
X = Feature
X[0:5]
```

Out[19]:

	order_amount	delivery_terms_days	customer_age	customer_gender	weekend	Bronze	Gold	Silver
0	402	7	36	0	0	False	True	False
1	735	7	30	1	0	False	False	True
2	570	7	37	0	1	False	True	False
3	406	30	29	0	1	True	False	False
4	371	15	37	0	0	False	True	False

What are our lables? Create Output Variable

In [20]:
```
y = df['order_status']
y[0:5]
d = {'NOT_DELIVERED':0,'DELIVERED' : 1}
y = y.map(d)
```

Normalize Data

Data Standardization give data zero mean and unit variance (technically should be done after train test split)

In [21]:
```
X= preprocessing.StandardScaler().fit(X).transform(X)
X[0:5]
```

Out[21]:
```
array([[-1.17179163, -1.06869132, -0.18071709, -0.998002  , -0.82674161,
        -0.75325833,  1.42167086, -0.66558354],
       [ 0.44736326, -1.06869132, -0.66651571,  1.002002  , -0.82674161,
        -0.75325833, -0.70339769,  1.50244101],
       [-0.35492069, -1.06869132, -0.09975065, -0.998002  ,  1.20956777,
        -0.75325833,  1.42167086, -0.66558354],
       [-1.15234232,  1.34133911, -0.74748215, -0.998002  ,  1.20956777,
         1.32756579, -0.70339769, -0.66558354],
       [-1.32252376, -0.23041987, -0.09975065, -0.998002  , -0.82674161,
        -0.75325833,  1.42167086, -0.66558354]])
```

Split the Data into Training and Testing Set

In [22]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2, random_state=4)
print ('Train set:', X_train.shape,  y_train.shape)
print ('Test set:', X_test.shape,  y_test.shape)
```

```
Train set: (800, 8) (800,)
Test set: (200, 8) (200,)
```

In [23]:
```
X_train_small = X_train[:200]
y_train_small = y_train[:200]
print('Reduced Train set:', X_train_small.shape, y_train_small.shape)
```

```
Reduced Train set: (200, 8) (200,)
```

Classification

Now, it is your turn, use the training set to build an accurate model. Then use the test set to report the accuracy of the model You should use the following algorithm:

- K Nearest Neighbor(KNN)
- Decision Tree
- Support Vector Machine
- Logistic Regression

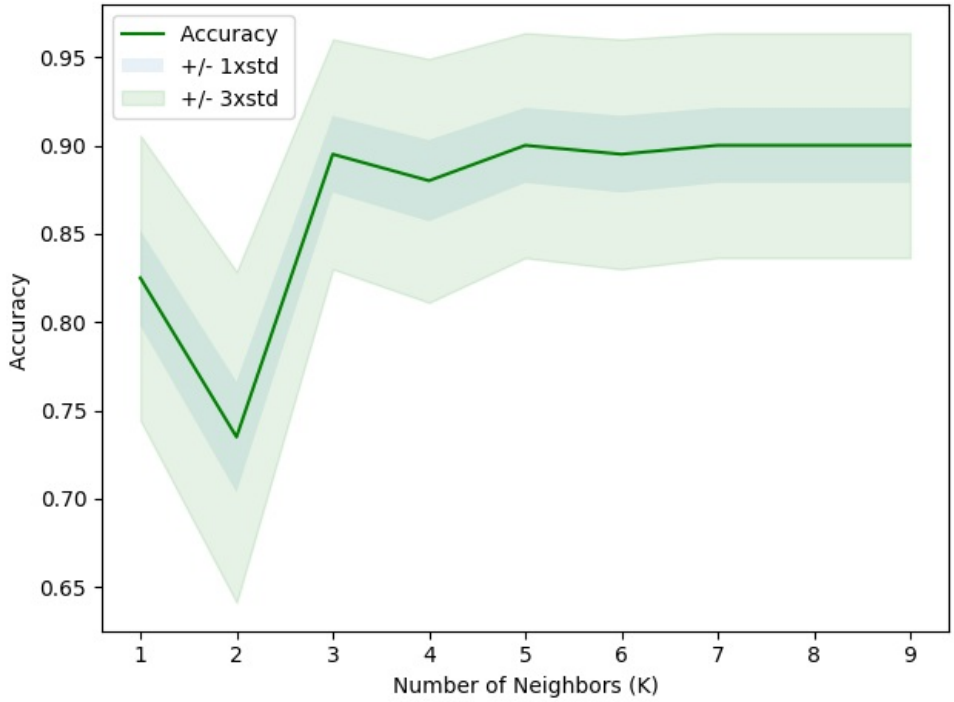
__ Notice:__

- You can go above and change the pre-processing, feature selection, feature-extraction, and so on, to make a better model.
- You should use either scikit-learn, Scipy or Numpy libraries for developing the classification algorithms.
- You should include the code of the algorithm in the following cells.

K Nearest Neighbor(KNN)

Notice: You should find the best k to build the model with the best accuracy.\

In [24]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
Ks = 10
mean_acc = np.zeros((Ks-1))
std_acc = np.zeros((Ks-1))

for n in range(1,Ks):
    knn1 = KNeighborsClassifier(n_neighbors = n).fit(X_train,y_train)
    yhat=knn1.predict(X_test)
    mean_acc[n-1] = metrics.accuracy_score(y_test, yhat)

    std_acc[n-1]=np.std(yhat==y_test)/np.sqrt(yhat.shape[0])

mean_acc
plt.plot(range(1,Ks),mean_acc,'g')
plt.fill_between(range(1,Ks),mean_acc - 1 * std_acc,mean_acc + 1 * std_acc, alpha=0.10)
plt.fill_between(range(1,Ks),mean_acc - 3 * std_acc,mean_acc + 3 * std_acc, alpha=0.10,color="green")
plt.legend(('Accuracy ', '+/- 1xstd','+/- 3xstd'))
plt.ylabel('Accuracy ')
plt.xlabel('Number of Neighbors (K)')
plt.tight_layout()
plt.show()
print( "The best accuracy was with", mean_acc.max(), "with k=", mean_acc.argmax()+1)
```

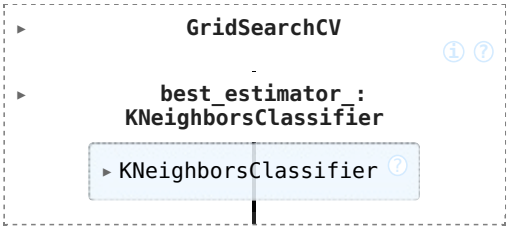



```
The best accuracy was with 0.9 with k= 5
```

Parameter Tunning using Grid Search Cv

In [25]:
```
from sklearn.model_selection import GridSearchCV
k_range = list(range(1, 31))
weight_options = ['uniform', 'distance']
pow = [1,2]
param_grid = dict(n_neighbors=k_range,
                  weights=weight_options,
                  p = pow)
knn_gs = KNeighborsClassifier()
grid_k = GridSearchCV(knn_gs,
                      param_grid,
                      cv=10,
                      scoring='accuracy')
grid_k.fit(X_train, y_train)
```

Out[25]:



In [26]:
```
print("Tuned Hyperparameters :", grid_k.best_params_)
print("Accuracy :",grid_k.best_score_)
```

```
Tuned Hyperparameters : {'n_neighbors': 7, 'p': 1, 'weights': 'uniform'}
Accuracy : 0.9
```

In [27]:
```
knn1 = KNeighborsClassifier(n_neighbors= 18, p = 1, weights = 'uniform')
```

In [28]:
```
knn1.fit(X_train,y_train)
```

Out[28]:

KNeighborsClassifier

```
KNeighborsClassifier(n_neighbors=18, p=1)
```

In [29]:
```
yhat = knn1.predict(X_test)
```

In [30]:
```
from sklearn.metrics import jaccard_score
from sklearn.metrics import f1_score
from sklearn.metrics import log_loss
from sklearn.metrics import accuracy_score
a1 = jaccard_score(y_test,yhat,pos_label=1)
b1 = f1_score(y_test, yhat, average='weighted')
c1 = accuracy_score(y_test, yhat)
print('The jaccard_score of the KNN for k = 7 classifier on train data is {:.2f}'.format(a1))
print('The F1-score of the KNN for k = 7 classifier on train data is {:.2f}'.format(b1))
print('The Accuracy_score of the KNN for k = 7 classifier on train data is {:.2f}'.format(c1))
```

```
The jaccard_score of the KNN for k = 7 classifier on train data is 0.90
The F1-score of the KNN for k = 7 classifier on train data is 0.85
The Accuracy_score of the KNN for k = 7 classifier on train data is 0.90
```

Decision Tree

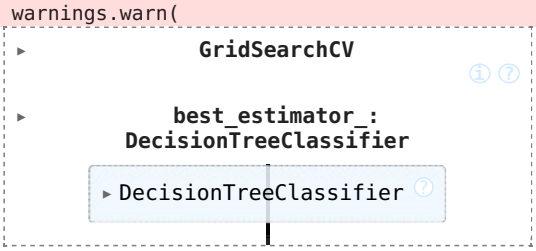
Parameter Tuning For Decision tree to find best tree

In [31]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
param_grid = {'criterion' :['gini', 'entropy'],
              'max_features': ['auto', 'sqrt', 'log2'],
              'ccp_alpha': [0.1, .01, .001],
              'max_depth' : [5, 6, 7, 8, 9],

             }
tree_clas = DecisionTreeClassifier(random_state=1024)
grid_search = GridSearchCV(estimator=tree_clas, param_grid=param_grid, cv=5, verbose=True)
grid_search.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 90 candidates, totalling 450 fits
```

```
C:\Users\Mohammed Meraj\AppData\Roaming\Python\Python312\site-packages\sklearn\model_selection\_validation.py:528: FitFailedWarning:
150 fits failed out of a total of 450.
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_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
150 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\Mohammed Meraj\AppData\Roaming\Python\Python312\site-packages\sklearn\model_selection\_validation.py", line 866, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\Mohammed Meraj\AppData\Roaming\Python\Python312\site-packages\sklearn\base.py", line 1382, in wrapper
    estimator._validate_params()
  File "C:\Users\Mohammed Meraj\AppData\Roaming\Python\Python312\site-packages\sklearn\base.py", line 436, in _validate_params
    validate_parameter_constraints(
  File "C:\Users\Mohammed Meraj\AppData\Roaming\Python\Python312\site-packages\sklearn\utils\_param_validation.py", line 98, in validate_parameter_constraints

    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'max_features' parameter of DecisionTreeClassifier must be an int in the range [1, inf), a float in the range (0.0, 1.0], a str among {'log2', 'sqrt'} or None. Got 'auto' instead.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Users\Mohammed Meraj\AppData\Roaming\Python\Python312\site-packages\sklearn\model_selection\_search.py:1108: UserWarning: One or more of the test scores are non-finite: [    nan 0.9     0.9         nan 0.9     0.9
nan 0.9     0.9
     nan 0.9     0.9         nan 0.9     0.9         nan 0.9     0.9
     nan 0.9     0.9         nan 0.9     0.9         nan 0.9     0.9
     nan 0.9     0.9         nan 0.9     0.9         nan 0.9     0.9
     nan 0.9     0.9         nan 0.9     0.9         nan 0.9     0.9
     nan 0.9     0.9         nan 0.9     0.9         nan 0.9     0.9
     nan 0.9     0.9         nan 0.9     0.9         nan 0.89375 0.88875
     nan 0.8925  0.88125     nan 0.875   0.86375     nan 0.87875 0.865
     nan 0.85625 0.855       nan 0.8925  0.8925      nan 0.88375 0.885
     nan 0.8625  0.88125     nan 0.86375 0.87625     nan 0.82625 0.8575 ]
  warnings.warn(
```

Out[31]:



Find the best parameters

In [32]:
```
print("Tuned Hyperparameters :", grid_search.best_params_)
print("Accuracy :",grid_search.best_score_)
```

```
Tuned Hyperparameters : {'ccp_alpha': 0.1, 'criterion': 'gini', 'max_depth': 5, 'max_features': 'sqrt'}
Accuracy : 0.9
```

In [33]:
```
Loan_Tree = grid_search.best_estimator_
print(Loan_Tree)
```

```
DecisionTreeClassifier(ccp_alpha=0.1, max_depth=5, max_features='sqrt',
                       random_state=1024)
```

Train the best model using Training Data

In [34]:
```
Loan_Tree.fit(X_train,y_train)
```

Out[34]:

DecisionTreeClassifier

```
DecisionTreeClassifier(ccp_alpha=0.1, max_depth=5, max_features='sqrt',
                       random_state=1024)
```

In [35]:
```
predTree = Loan_Tree.predict(X_test)
```

In [36]:
```
a2 = jaccard_score(y_test, predTree,pos_label=1)
b2 = f1_score(y_test, predTree, average='weighted')
c2 = accuracy_score(y_test, predTree)
print("The accuraccy of (Loan_ tree) DecisionTrees's {:.2f} ".format(c2))
```

```
print('The jaccard_score of the (Loan_ tree) DecisionTrees classifier on train data is {:.2f}'.format(a2))
print('The F1-score of the (Loan_ tree) DecisionTrees classifier on train data is {:.2f}'.format(b2))
```

```
The accuraccy of (Loan_ tree) DecisionTrees's 0.90
The jaccard_score of the (Loan_ tree) DecisionTrees classifier on train data is 0.90
The F1-score of the (Loan_ tree) DecisionTrees classifier on train data is 0.85
```

Support Vector Machine

Parameter Tunning For SVM using GridSerachCV

In [37]:
```
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV

param_grid = {'C': [0.1,1, 10, 100],
              'gamma': [1,0.1,0.01,0.001],
              'kernel': ['rbf', 'poly', 'sigmoid']}
grid_s = GridSearchCV(SVC(),param_grid,refit=True,verbose=2)
grid_s.fit(X_train_small,y_train_small)
```

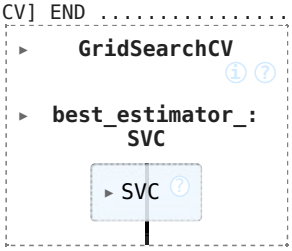
```
Fitting 5 folds for each of 48 candidates, totalling 240 fits
[CV] END .....................C=0.1, gamma=1, kernel=rbf; total time=   0.0s
[CV] END .....................C=0.1, gamma=1, kernel=rbf; total time=   0.0s
[CV] END .....................C=0.1, gamma=1, kernel=rbf; total time=   0.0s
[CV] END .....................C=0.1, gamma=1, kernel=rbf; total time=   0.0s
[CV] END .....................C=0.1, gamma=1, kernel=rbf; total time=   0.0s
[CV] END .....................C=0.1, gamma=1, kernel=poly; total time=   0.0s
[CV] END .....................C=0.1, gamma=1, kernel=poly; total time=   0.0s
[CV] END .....................C=0.1, gamma=1, kernel=poly; total time=   0.0s
[CV] END .....................C=0.1, gamma=1, kernel=poly; total time=   0.0s
[CV] END .....................C=0.1, gamma=1, kernel=poly; total time=   0.0s
[CV] END ....................C=0.1, gamma=1, kernel=sigmoid; total time=   0.0s
[CV] END ....................C=0.1, gamma=1, kernel=sigmoid; total time=   0.0s
[CV] END ....................C=0.1, gamma=1, kernel=sigmoid; total time=   0.0s
[CV] END ....................C=0.1, gamma=1, kernel=sigmoid; total time=   0.0s
[CV] END ....................C=0.1, gamma=1, kernel=sigmoid; total time=   0.0s
[CV] END ...................C=0.1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END ...................C=0.1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END ...................C=0.1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END ...................C=0.1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END ...................C=0.1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END ...................C=0.1, gamma=0.1, kernel=poly; total time=   0.0s
[CV] END ...................C=0.1, gamma=0.1, kernel=poly; total time=   0.0s
[CV] END ...................C=0.1, gamma=0.1, kernel=poly; total time=   0.0s
[CV] END ...................C=0.1, gamma=0.1, kernel=poly; total time=   0.0s
[CV] END ...................C=0.1, gamma=0.1, kernel=poly; total time=   0.0s
[CV] END ................C=0.1, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END ................C=0.1, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END ................C=0.1, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END ................C=0.1, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END ................C=0.1, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END ..................C=0.1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END ..................C=0.1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END ..................C=0.1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END ..................C=0.1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END ..................C=0.1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END ..................C=0.1, gamma=0.01, kernel=poly; total time=   0.0s
[CV] END ..................C=0.1, gamma=0.01, kernel=poly; total time=   0.0s
[CV] END ..................C=0.1, gamma=0.01, kernel=poly; total time=   0.0s
[CV] END ..................C=0.1, gamma=0.01, kernel=poly; total time=   0.0s
[CV] END ..................C=0.1, gamma=0.01, kernel=poly; total time=   0.0s
[CV] END ..............C=0.1, gamma=0.01, kernel=sigmoid; total time=   0.0s
[CV] END ..............C=0.1, gamma=0.01, kernel=sigmoid; total time=   0.0s
[CV] END ..............C=0.1, gamma=0.01, kernel=sigmoid; total time=   0.0s
[CV] END ..............C=0.1, gamma=0.01, kernel=sigmoid; total time=   0.0s
[CV] END ..............C=0.1, gamma=0.01, kernel=sigmoid; total time=   0.0s
[CV] END .................C=0.1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END .................C=0.1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END .................C=0.1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END .................C=0.1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END .................C=0.1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END .................C=0.1, gamma=0.001, kernel=poly; total time=   0.0s
[CV] END .................C=0.1, gamma=0.001, kernel=poly; total time=   0.0s
[CV] END .................C=0.1, gamma=0.001, kernel=poly; total time=   0.0s
[CV] END .................C=0.1, gamma=0.001, kernel=poly; total time=   0.0s
[CV] END .................C=0.1, gamma=0.001, kernel=poly; total time=   0.0s
[CV] END ............C=0.1, gamma=0.001, kernel=sigmoid; total time=   0.0s
[CV] END ............C=0.1, gamma=0.001, kernel=sigmoid; total time=   0.0s
[CV] END ............C=0.1, gamma=0.001, kernel=sigmoid; total time=   0.0s
[CV] END ............C=0.1, gamma=0.001, kernel=sigmoid; total time=   0.0s
[CV] END ............C=0.1, gamma=0.001, kernel=sigmoid; total time=   0.0s
```

```
[CV] END .........................C=1, gamma=1, kernel=rbf; total time=   0.0s
[CV] END .........................C=1, gamma=1, kernel=rbf; total time=   0.0s
[CV] END .........................C=1, gamma=1, kernel=rbf; total time=   0.0s
[CV] END .........................C=1, gamma=1, kernel=rbf; total time=   0.0s
[CV] END .........................C=1, gamma=1, kernel=rbf; total time=   0.0s
[CV] END ........................C=1, gamma=1, kernel=poly; total time=   0.0s
[CV] END ........................C=1, gamma=1, kernel=poly; total time=   0.0s
[CV] END ........................C=1, gamma=1, kernel=poly; total time=   0.0s
[CV] END ........................C=1, gamma=1, kernel=poly; total time=   0.0s
[CV] END ........................C=1, gamma=1, kernel=poly; total time=   0.0s
[CV] END .......................C=1, gamma=1, kernel=sigmoid; total time=   0.0s
[CV] END .......................C=1, gamma=1, kernel=sigmoid; total time=   0.0s
[CV] END .......................C=1, gamma=1, kernel=sigmoid; total time=   0.0s
[CV] END .......................C=1, gamma=1, kernel=sigmoid; total time=   0.0s
[CV] END .......................C=1, gamma=1, kernel=sigmoid; total time=   0.0s
[CV] END .........................C=1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END .........................C=1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END .........................C=1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END .........................C=1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END .........................C=1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END ........................C=1, gamma=0.1, kernel=poly; total time=   0.0s
[CV] END ........................C=1, gamma=0.1, kernel=poly; total time=   0.0s
[CV] END ........................C=1, gamma=0.1, kernel=poly; total time=   0.0s
[CV] END ........................C=1, gamma=0.1, kernel=poly; total time=   0.0s
[CV] END ........................C=1, gamma=0.1, kernel=poly; total time=   0.0s
[CV] END .......................C=1, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END .......................C=1, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END .......................C=1, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END .......................C=1, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END .......................C=1, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END ........................C=1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END ........................C=1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END ........................C=1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END ........................C=1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END ........................C=1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END .......................C=1, gamma=0.01, kernel=poly; total time=   0.0s
[CV] END .......................C=1, gamma=0.01, kernel=poly; total time=   0.0s
[CV] END .......................C=1, gamma=0.01, kernel=poly; total time=   0.0s
[CV] END .......................C=1, gamma=0.01, kernel=poly; total time=   0.0s
[CV] END .......................C=1, gamma=0.01, kernel=poly; total time=   0.0s
[CV] END ......................C=1, gamma=0.01, kernel=sigmoid; total time=   0.0s
[CV] END ......................C=1, gamma=0.01, kernel=sigmoid; total time=   0.0s
[CV] END ......................C=1, gamma=0.01, kernel=sigmoid; total time=   0.0s
[CV] END ......................C=1, gamma=0.01, kernel=sigmoid; total time=   0.0s
[CV] END ......................C=1, gamma=0.01, kernel=sigmoid; total time=   0.0s
[CV] END .......................C=1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END .......................C=1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END .......................C=1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END .......................C=1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END .......................C=1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END ......................C=1, gamma=0.001, kernel=poly; total time=   0.0s
[CV] END ......................C=1, gamma=0.001, kernel=poly; total time=   0.0s
[CV] END ......................C=1, gamma=0.001, kernel=poly; total time=   0.0s
[CV] END ......................C=1, gamma=0.001, kernel=poly; total time=   0.0s
[CV] END ......................C=1, gamma=0.001, kernel=poly; total time=   0.0s
[CV] END .....................C=1, gamma=0.001, kernel=sigmoid; total time=   0.0s
[CV] END .....................C=1, gamma=0.001, kernel=sigmoid; total time=   0.0s
[CV] END .....................C=1, gamma=0.001, kernel=sigmoid; total time=   0.0s
[CV] END .....................C=1, gamma=0.001, kernel=sigmoid; total time=   0.0s
[CV] END .....................C=1, gamma=0.001, kernel=sigmoid; total time=   0.0s
[CV] END ........................C=10, gamma=1, kernel=rbf; total time=   0.0s
[CV] END ........................C=10, gamma=1, kernel=rbf; total time=   0.0s
[CV] END ........................C=10, gamma=1, kernel=rbf; total time=   0.0s
[CV] END ........................C=10, gamma=1, kernel=rbf; total time=   0.0s
[CV] END ........................C=10, gamma=1, kernel=rbf; total time=   0.0s
[CV] END .......................C=10, gamma=1, kernel=poly; total time=   0.0s
[CV] END .......................C=10, gamma=1, kernel=poly; total time=   0.0s
[CV] END .......................C=10, gamma=1, kernel=poly; total time=   0.0s
[CV] END .......................C=10, gamma=1, kernel=poly; total time=   0.0s
[CV] END .......................C=10, gamma=1, kernel=poly; total time=   0.0s
[CV] END .....................C=10, gamma=1, kernel=sigmoid; total time=   0.0s
[CV] END .....................C=10, gamma=1, kernel=sigmoid; total time=   0.0s
[CV] END .....................C=10, gamma=1, kernel=sigmoid; total time=   0.0s
[CV] END .....................C=10, gamma=1, kernel=sigmoid; total time=   0.0s
[CV] END .....................C=10, gamma=1, kernel=sigmoid; total time=   0.0s
[CV] END .......................C=10, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END .......................C=10, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END .......................C=10, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END .......................C=10, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END .......................C=10, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END ......................C=10, gamma=0.1, kernel=poly; total time=   0.0s
[CV] END ......................C=10, gamma=0.1, kernel=poly; total time=   0.0s
[CV] END ......................C=10, gamma=0.1, kernel=poly; total time=   0.0s
```

```
[CV] END ......................C=10, gamma=0.1, kernel=poly; total time=   0.0s
[CV] END ......................C=10, gamma=0.1, kernel=poly; total time=   0.0s
[CV] END ......................C=10, gamma=0.1, kernel=poly; total time=   0.0s
[CV] END ......................C=10, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END ......................C=10, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END ......................C=10, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END ......................C=10, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END .......................C=10, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END .......................C=10, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END .......................C=10, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END .......................C=10, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END .......................C=10, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END .......................C=10, gamma=0.01, kernel=poly; total time=   0.0s
[CV] END .......................C=10, gamma=0.01, kernel=poly; total time=   0.0s
[CV] END .......................C=10, gamma=0.01, kernel=poly; total time=   0.0s
[CV] END .......................C=10, gamma=0.01, kernel=poly; total time=   0.0s
[CV] END .......................C=10, gamma=0.01, kernel=poly; total time=   0.0s
[CV] END ....................C=10, gamma=0.01, kernel=sigmoid; total time=   0.0s
[CV] END ....................C=10, gamma=0.01, kernel=sigmoid; total time=   0.0s
[CV] END ....................C=10, gamma=0.01, kernel=sigmoid; total time=   0.0s
[CV] END ....................C=10, gamma=0.01, kernel=sigmoid; total time=   0.0s
[CV] END ....................C=10, gamma=0.01, kernel=sigmoid; total time=   0.0s
[CV] END ......................C=10, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END ......................C=10, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END ......................C=10, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END ......................C=10, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END ......................C=10, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END ......................C=10, gamma=0.001, kernel=poly; total time=   0.0s
[CV] END ......................C=10, gamma=0.001, kernel=poly; total time=   0.0s
[CV] END ......................C=10, gamma=0.001, kernel=poly; total time=   0.0s
[CV] END ......................C=10, gamma=0.001, kernel=poly; total time=   0.0s
[CV] END ......................C=10, gamma=0.001, kernel=poly; total time=   0.0s
[CV] END ....................C=10, gamma=0.001, kernel=sigmoid; total time=   0.0s
[CV] END ....................C=10, gamma=0.001, kernel=sigmoid; total time=   0.0s
[CV] END ....................C=10, gamma=0.001, kernel=sigmoid; total time=   0.0s
[CV] END ....................C=10, gamma=0.001, kernel=sigmoid; total time=   0.0s
[CV] END ....................C=10, gamma=0.001, kernel=sigmoid; total time=   0.0s
[CV] END .........................C=100, gamma=1, kernel=rbf; total time=   0.0s
[CV] END .........................C=100, gamma=1, kernel=rbf; total time=   0.0s
[CV] END .........................C=100, gamma=1, kernel=rbf; total time=   0.0s
[CV] END .........................C=100, gamma=1, kernel=rbf; total time=   0.0s
[CV] END .........................C=100, gamma=1, kernel=rbf; total time=   0.0s
[CV] END .........................C=100, gamma=1, kernel=poly; total time=   0.0s
[CV] END .........................C=100, gamma=1, kernel=poly; total time=   0.0s
[CV] END .........................C=100, gamma=1, kernel=poly; total time=   0.0s
[CV] END .........................C=100, gamma=1, kernel=poly; total time=   0.0s
[CV] END .........................C=100, gamma=1, kernel=poly; total time=   0.0s
[CV] END .....................C=100, gamma=1, kernel=sigmoid; total time=   0.0s
[CV] END .....................C=100, gamma=1, kernel=sigmoid; total time=   0.0s
[CV] END .....................C=100, gamma=1, kernel=sigmoid; total time=   0.0s
[CV] END .....................C=100, gamma=1, kernel=sigmoid; total time=   0.0s
[CV] END .....................C=100, gamma=1, kernel=sigmoid; total time=   0.0s
[CV] END .......................C=100, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END .......................C=100, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END .......................C=100, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END .......................C=100, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END .......................C=100, gamma=0.1, kernel=rbf; total time=   0.0s
[CV] END .......................C=100, gamma=0.1, kernel=poly; total time=   0.0s
[CV] END .......................C=100, gamma=0.1, kernel=poly; total time=   0.0s
[CV] END .......................C=100, gamma=0.1, kernel=poly; total time=   0.0s
[CV] END .......................C=100, gamma=0.1, kernel=poly; total time=   0.0s
[CV] END .......................C=100, gamma=0.1, kernel=poly; total time=   0.0s
[CV] END .....................C=100, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END .....................C=100, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END .....................C=100, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END .....................C=100, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END .....................C=100, gamma=0.1, kernel=sigmoid; total time=   0.0s
[CV] END ......................C=100, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END ......................C=100, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END ......................C=100, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END ......................C=100, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END ......................C=100, gamma=0.01, kernel=rbf; total time=   0.0s
[CV] END ......................C=100, gamma=0.01, kernel=poly; total time=   0.0s
[CV] END ......................C=100, gamma=0.01, kernel=poly; total time=   0.0s
[CV] END ......................C=100, gamma=0.01, kernel=poly; total time=   0.0s
[CV] END ......................C=100, gamma=0.01, kernel=poly; total time=   0.0s
[CV] END ......................C=100, gamma=0.01, kernel=poly; total time=   0.0s
[CV] END .....................C=100, gamma=0.01, kernel=sigmoid; total time=   0.0s
[CV] END .....................C=100, gamma=0.01, kernel=sigmoid; total time=   0.0s
[CV] END .....................C=100, gamma=0.01, kernel=sigmoid; total time=   0.0s
[CV] END .....................C=100, gamma=0.01, kernel=sigmoid; total time=   0.0s
[CV] END .....................C=100, gamma=0.01, kernel=sigmoid; total time=   0.0s
[CV] END ......................C=100, gamma=0.001, kernel=rbf; total time=   0.0s
```

```
[CV] END .....................C=100, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END .....................C=100, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END .....................C=100, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END .....................C=100, gamma=0.001, kernel=rbf; total time=   0.0s
[CV] END ....................C=100, gamma=0.001, kernel=poly; total time=   0.0s
[CV] END ....................C=100, gamma=0.001, kernel=poly; total time=   0.0s
[CV] END ....................C=100, gamma=0.001, kernel=poly; total time=   0.0s
[CV] END ....................C=100, gamma=0.001, kernel=poly; total time=   0.0s
[CV] END ....................C=100, gamma=0.001, kernel=poly; total time=   0.0s
[CV] END .................C=100, gamma=0.001, kernel=sigmoid; total time=   0.0s
[CV] END .................C=100, gamma=0.001, kernel=sigmoid; total time=   0.0s
[CV] END .................C=100, gamma=0.001, kernel=sigmoid; total time=   0.0s
[CV] END .................C=100, gamma=0.001, kernel=sigmoid; total time=   0.0s
[CV] END .................C=100, gamma=0.001, kernel=sigmoid; total time=   0.0s
```

Out[37]:



In [38]:
```
print("Tuned Hyperparameters :", grid_s.best_params_)
print("Accuracy :",grid_s.best_score_)
```

```
Tuned Hyperparameters : {'C': 0.1, 'gamma': 1, 'kernel': 'rbf'}
Accuracy : 0.8949999999999999
```

In [39]:
```
svm = SVC(probability=True,C=0.1, gamma=1, kernel='sigmoid')
print(svm)
```

```
SVC(C=0.1, gamma=1, kernel='sigmoid', probability=True)
```

In [40]:
```
svm.fit(X_train, y_train)
```

Out[40]:

```
SVC
SVC(C=0.1, gamma=1, kernel='sigmoid', probability=True)
```

In [41]:
```
yhat_s = svm.predict(X_test)
```

In [42]:
```
a3 = jaccard_score(y_test, yhat_s,pos_label=1)
b3 = f1_score(y_test, yhat_s, average='weighted')
c3 = accuracy_score(y_test, yhat_s)
print('The accuracy of the svm classifier on training data is {:.2f} out of 1'.format(svm.score(X_train, y_trai
print('The accuracy of the svm classifier on test data is {:.2f} out of 1'.format(svm.score(X_test, y_test)))
print('The jaccard_score of the SVM classifier on train data is {:.2f}'.format(a3))
print('The F1-score of the SVM classifier on train data is {:.2f}'.format(b3))
print('The accuracy-score of the SVM classifier on train data is {:.2f}'.format(c3))
```

```
The accuracy of the svm classifier on training data is 0.88 out of 1
The accuracy of the svm classifier on test data is 0.88 out of 1
The jaccard_score of the SVM classifier on train data is 0.88
The F1-score of the SVM classifier on train data is 0.84
The accuracy-score of the SVM classifier on train data is 0.88
```

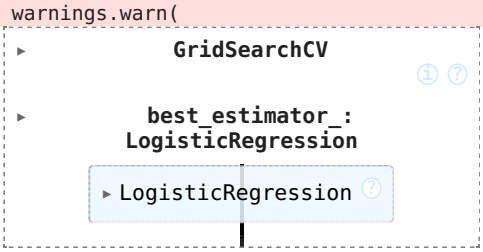
Logistic Regression

Parameter Tunning using Grid Serch CV

In [43]:
```
from sklearn.linear_model import LogisticRegression
parameters = {
    'penalty' : ['l1','l2'],
    'C'       : np.logspace(-3,3,7),
    'solver'  : ['newton-cg', 'lbfgs', 'liblinear'],
}
logreg = LogisticRegression()
clf = GridSearchCV(logreg,                  # model
                   param_grid = parameters,   # hyperparameters
                   scoring='accuracy',        # metric for scoring
                   cv=10)                     # number of folds
clf.fit(X_train,y_train)
```

```
C:\Users\Mohammed Meraj\AppData\Roaming\Python\Python312\site-packages\sklearn\model_selection\_validation.py:528: FitFailedWarning:
140 fits failed out of a total of 420.
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_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
70 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\Mohammed Meraj\AppData\Roaming\Python\Python312\site-packages\sklearn\model_selection\_validation.py", line 866, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\Mohammed Meraj\AppData\Roaming\Python\Python312\site-packages\sklearn\base.py", line 1389, in wrapper
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Mohammed Meraj\AppData\Roaming\Python\Python312\site-packages\sklearn\linear_model\_logistic.py", line 1193, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Mohammed Meraj\AppData\Roaming\Python\Python312\site-packages\sklearn\linear_model\_logistic.py", line 63, in _check_solver
    raise ValueError(
ValueError: Solver newton-cg supports only 'l2' or None penalties, got l1 penalty.

--------------------------------------------------------------------------------
70 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\Mohammed Meraj\AppData\Roaming\Python\Python312\site-packages\sklearn\model_selection\_validation.py", line 866, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\Mohammed Meraj\AppData\Roaming\Python\Python312\site-packages\sklearn\base.py", line 1389, in wrapper
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Mohammed Meraj\AppData\Roaming\Python\Python312\site-packages\sklearn\linear_model\_logistic.py", line 1193, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Mohammed Meraj\AppData\Roaming\Python\Python312\site-packages\sklearn\linear_model\_logistic.py", line 63, in _check_solver
    raise ValueError(
ValueError: Solver lbfgs supports only 'l2' or None penalties, got l1 penalty.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Users\Mohammed Meraj\AppData\Roaming\Python\Python312\site-packages\sklearn\model_selection\_search.py:1108: UserWarning: One or more of the test scores are non-finite: [nan nan 0.1 0.9 0.9 0.9 nan nan 0.9 0.9 0.9 0.9 nan nan 0.9 0.9 0.9 0.9
 nan nan 0.9 0.9 0.9 0.9 nan nan 0.9 0.9 0.9 0.9 nan nan 0.9 0.9 0.9 0.9
 nan nan 0.9 0.9 0.9 0.9]
  warnings.warn(
```

Out[43]:

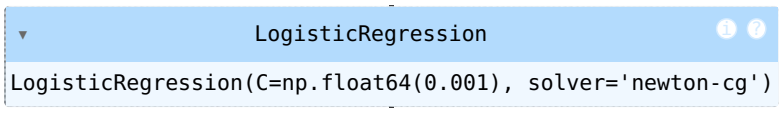


In [44]:
```
print("Tuned Hyperparameters :", clf.best_params_)
print("Accuracy :",clf.best_score_)
```

```
Tuned Hyperparameters : {'C': np.float64(0.001), 'penalty': 'l2', 'solver': 'newton-cg'}
Accuracy : 0.9
```

In [45]:
```
log_reg = clf.best_estimator_
log_reg.fit(X_train,y_train)
```

Out[45]:



In [46]:
```
yhat_l = log_reg.predict(X_test)
```

In [47]:
```
a4 = jaccard_score(y_test, yhat_l,pos_label=1)
b4 = f1_score(y_test, yhat_l, average='weighted')
c4 = accuracy_score(y_test, yhat_l)

print('The jaccard_score of the logistic regression classifier on train data is {:.2f}'.format(a4))
```

```
print('The F1-score of the logistic regression classifier on train data is {:.2f}'.format(b4))
print('The accuracy_score of the logistic regression classifier on train data is {:.2f}'.format(c4))
```

```
The jaccard_score of the logistic regression classifier on train data is 0.90
The F1-score of the logistic regression classifier on train data is 0.85
The accuracy_score of the logistic regression classifier on train data is 0.90
```

Model Evaluation

In [48]:
```
result_df=pd.DataFrame({'Model':['KNN','Decision Tree','SVM','Logistic Regression'],
                        'Jaccard Score' : [a1,a2,a3,a4],
                        'F1 Score' : [b1,b2,b3,b4],
                        'Accuracy Score':[c1,c2,c3,c4]})
```

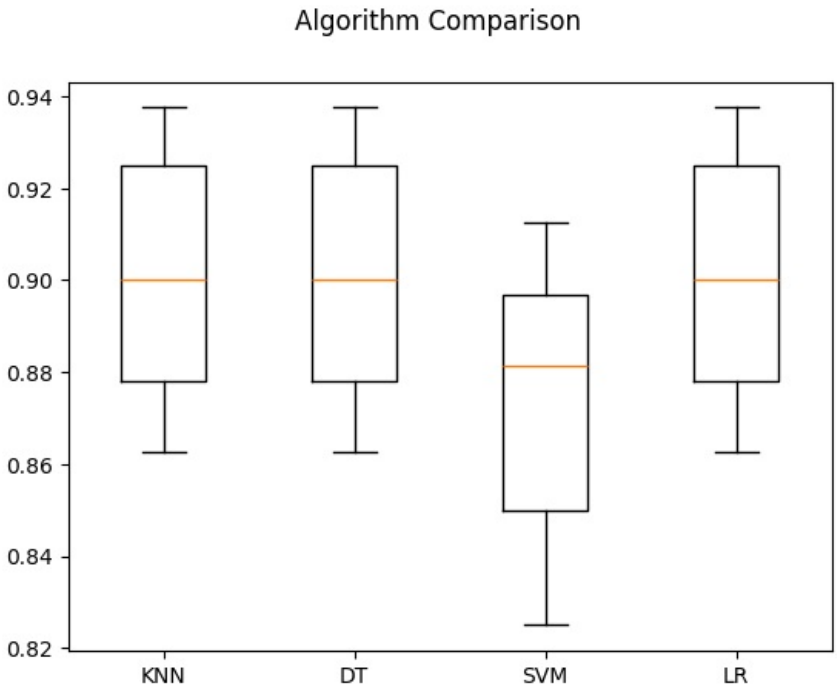
In [49]:
```
print(result_df)
```

```
                 Model  Jaccard Score  F1 Score  Accuracy Score
0                  KNN           0.90  0.852632            0.90
1        Decision Tree           0.90  0.852632            0.90
2                  SVM           0.88  0.842553            0.88
3  Logistic Regression           0.90  0.852632            0.90
```

using K-fold cross validation

In [50]:
```
from sklearn import model_selection
# prepare configuration for cross validation test harness
seed = 42
# prepare models
models = []
models.append(('KNN', knn1))
models.append(('DT', Loan_Tree ))
models.append(('SVM', svm))
models.append(('LR', log_reg))
# evaluate each model in turn
results = []
names = []
scoring = 'accuracy'
for name,model in models:
    kfold = model_selection.KFold(n_splits=10,random_state=seed,shuffle=True)
    cv_results = model_selection.cross_val_score(model,X_train,y_train,cv=kfold,scoring=scoring)
    results.append(cv_results)
    names.append(name)

result_mean =[]
result_std = []
i = 0
for name,model in models:
    msg = "%s: %f (%f)" % (name,results[i].mean(),
                           results[i].std())
    result_mean.append(results[i].mean()*100)
    result_std.append(results[i].std())
    i = i+1
    print(msg)
# boxplot algorithm comparison
fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

```
KNN: 0.900000 (0.026220)
DT: 0.900000 (0.026220)
SVM: 0.873750 (0.027071)
LR: 0.900000 (0.026220)
```



Ploting ROC_AUC Curce

In [51]:
```
#d = {'COLLECTION':0,'PAIDOFF' : 1}
#y_test = y_test.map(d)
```

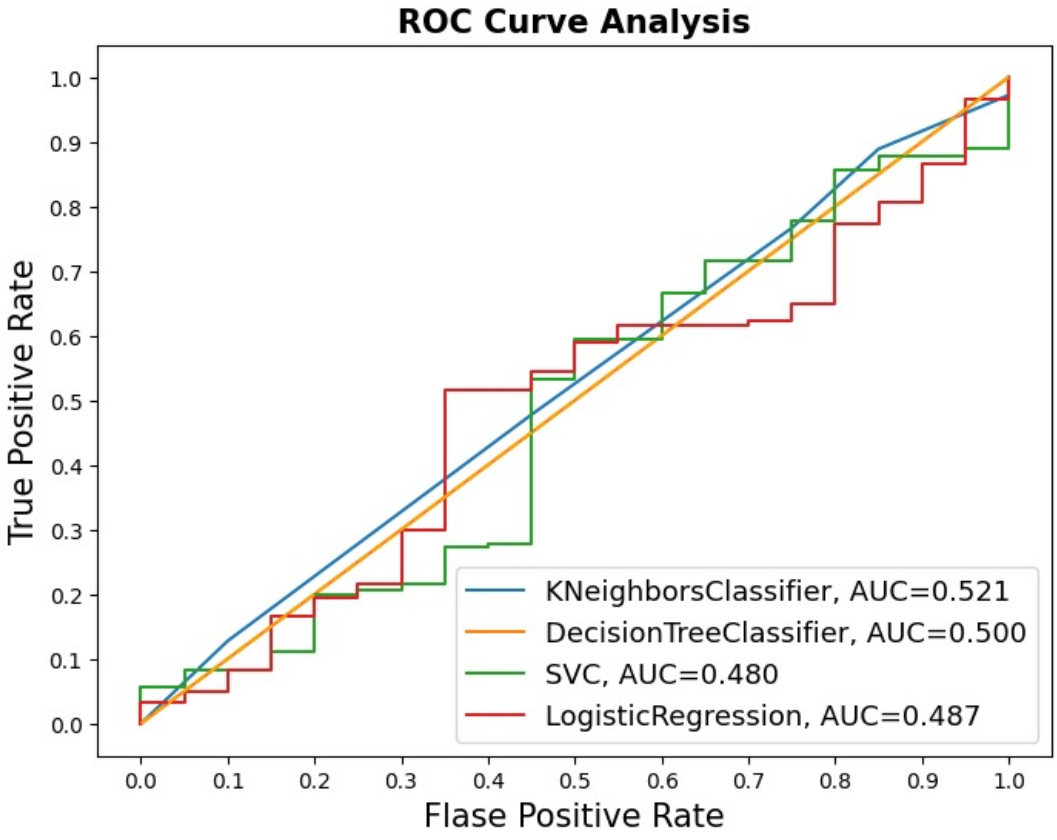
In [54]:
```
from sklearn.metrics import roc_curve, roc_auc_score

# Instantiate the classfiers and make a list
classifiers = [knn1,
               Loan_Tree,
               svm,
               log_reg]
model = ['KNN',
         'Decision Tree',
         'SVM',
         'Logistic Regression']
# Define a result table as a DataFrame
result_table = pd.DataFrame(columns=['model', 'fpr','tpr','auc'])

# Train the models and record the results
for cls in classifiers:
    model = cls.fit(X_train, y_train)
    yproba = model.predict_proba(X_test)[::,1]
    fpr, tpr, _ = roc_curve(y_test,  yproba)
    auc = roc_auc_score(y_test, yproba)

    result_table.loc[len(result_table)] = [cls.__class__.__name__, fpr, tpr, auc]

# Set name of the classifiers as index labels
result_table.set_index('model', inplace=True)
```

In [55]:
```
print(result_table)
```

```
                                                                fpr  \
model
KNeighborsClassifier              [0.0, 0.1, 0.45, 0.75, 0.85, 1.0, 1.0]
DecisionTreeClassifier                                     [0.0, 1.0]
SVC                    [0.0, 0.0, 0.0, 0.05, 0.05, 0.15, 0.15, 0.2, 0...
LogisticRegression     [0.0, 0.0, 0.0, 0.05, 0.05, 0.1, 0.1, 0.15, 0....

                                                                tpr  \
model
KNeighborsClassifier    [0.0, 0.12777777777777777, 0.4777777777777778,...
DecisionTreeClassifier                                     [0.0, 1.0]
SVC                    [0.0, 0.005555555555555556, 0.0555555555555555...
LogisticRegression     [0.0, 0.005555555555555556, 0.0333333333333333...

                            auc
model
KNeighborsClassifier    0.521389
DecisionTreeClassifier  0.500000
SVC                     0.480278
LogisticRegression      0.487222
```

Plot The Figure

In [56]:
```
fig = plt.figure(figsize=(8,6))

for i in result_table.index:
    plt.plot(result_table.loc[i]['fpr'],
             result_table.loc[i]['tpr'],
             label="{}, AUC={:.3f}".format(i, result_table.loc[i]['auc']))

plt.plot([0,1], [0,1], color='orange', linestyle='--')

plt.xticks(np.arange(0.0, 1.1, step=0.1))
plt.xlabel("Flase Positive Rate", fontsize=15)

plt.yticks(np.arange(0.0, 1.1, step=0.1))
plt.ylabel("True Positive Rate", fontsize=15)

plt.title('ROC Curve Analysis', fontweight='bold', fontsize=15)
plt.legend(prop={'size':13}, loc='lower right')

plt.show()
```



In [ ]: