APPENDIX

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score,confusion_matrix
%matplotlib inline
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files diabetes_data_upload.csv

- **diabetes_data_upload.csv**(text/csv) - 34682 bytes, last modified: 5/19/2022 - 100% done

```
Saving diabetes_data_upload.csv to diabetes_data_upload (1).csv
```

```
dataset=pd.read_csv("diabetes_data_upload.csv")
dataset
```

	Age	Gender	Polyuria	Polydipsia	sudden weight loss	weakness	Polyphagia	Genital thrush	visual blurring
0	40	1	0	1	0	1	0	0	0
1	58	1	0	0	0	1	0	0	1
2	41	1	1	0	0	1	1	0	0
3	45	1	0	0	1	1	1	1	0
4	60	1	1	1	1	1	1	0	1
...	...	...	...	...	...	...	...	...	...
515	39	0	1	1	1	0	1	0	0
516	48	0	1	1	1	1	1	0	0
517	58	0	1	1	1	1	1	0	1
518	32	0	0	0	0	1	0	0	1
519	42	1	0	0	0	0	0	0	0

520 rows × 17 columns

```
dataset.shape
```

```
(520, 17)
```

```
dataset.describe()
```

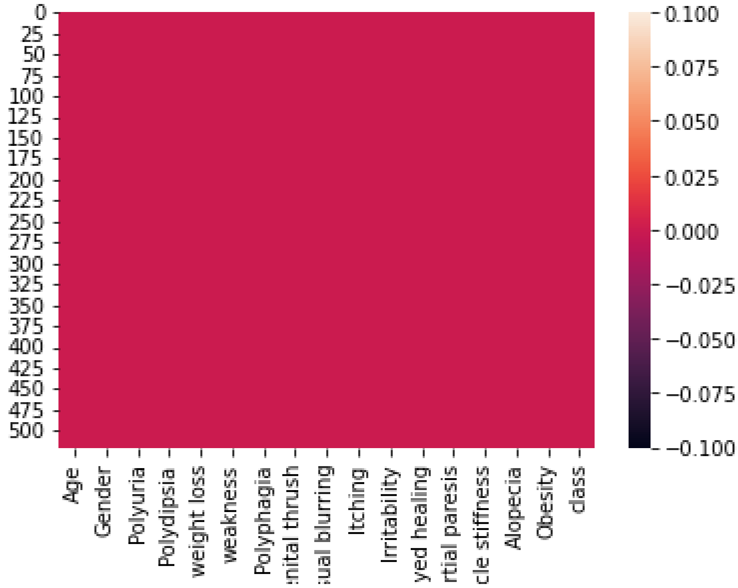
	Age
count	520.000000
mean	48.028846
std	12.151466
min	16.000000
25%	39.000000
50%	47.500000
75%	57.000000
max	90.000000

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 520 entries, 0 to 519
Data columns (total 17 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   Age                 520 non-null    int64 
 1   Gender              520 non-null    object
 2   Polyuria            520 non-null    object
 3   Polydipsia          520 non-null    object
 4   sudden weight loss  520 non-null    object
 5   weakness            520 non-null    object
 6   Polyphagia          520 non-null    object
 7   Genital thrush      520 non-null    object
 8   visual blurring     520 non-null    object
 9   Itching             520 non-null    object
 10  Irritability        520 non-null    object
 11  delayed healing     520 non-null    object
 12  partial paresis     520 non-null    object
 13  muscle stiffness    520 non-null    object
 14  Alopecia            520 non-null    object
 15  Obesity             520 non-null    object
 16  class               520 non-null    object
dtypes: int64(1), object(16)
memory usage: 69.2+ KB
```

```
sns.heatmap(dataset.isnull())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2d16ef0210>
```




```
dataset['class'].value_counts()
```

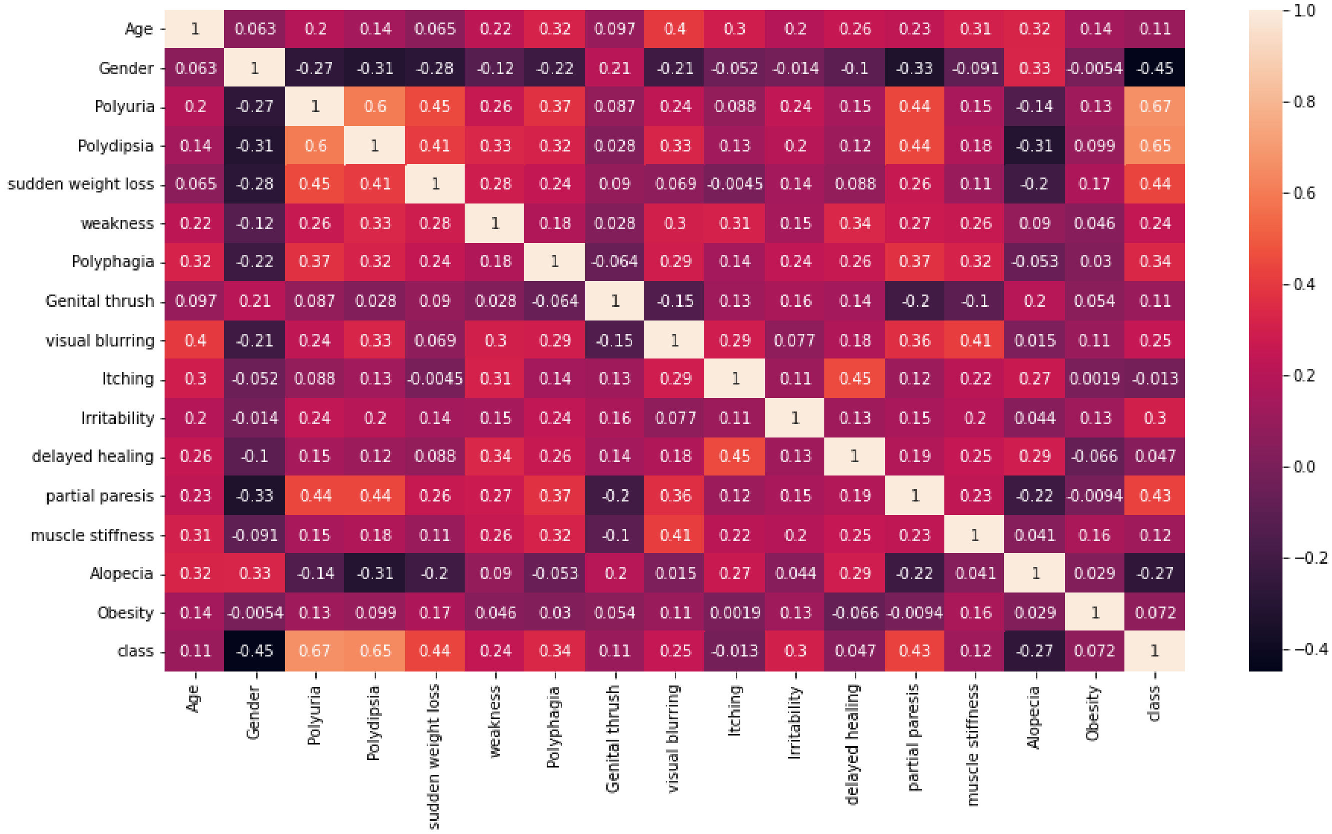
```
Positive    320
Negative    200
Name: class, dtype: int64
```

```
dataset['Gender'] = dataset['Gender'].map({'Male':1,'Female':0})
dataset['class'] = dataset['class'].map({'Positive':1,'Negative':0})
dataset['Polyuria'] = dataset['Polyuria'].map({'Yes':1,'No':0})
dataset['Polydipsia'] = dataset['Polydipsia'].map({'Yes':1,'No':0})
dataset['sudden weight loss'] = dataset['sudden weight loss'].map({'Yes':1,'No':0})
dataset['weakness'] = dataset['weakness'].map({'Yes':1,'No':0})
dataset['Polyphagia'] = dataset['Polyphagia'].map({'Yes':1,'No':0})
dataset['Genital thrush'] = dataset['Genital thrush'].map({'Yes':1,'No':0})
dataset['visual blurring'] = dataset['visual blurring'].map({'Yes':1,'No':0})
dataset['Itching'] = dataset['Itching'].map({'Yes':1,'No':0})
dataset['Irritability'] = dataset['Irritability'].map({'Yes':1,'No':0})
dataset['delayed healing'] = dataset['delayed healing'].map({'Yes':1,'No':0})
dataset['partial paresis'] = dataset['partial paresis'].map({'Yes':1,'No':0})
dataset['muscle stiffness'] = dataset['muscle stiffness'].map({'Yes':1,'No':0})
dataset['Alopecia'] = dataset['Alopecia'].map({'Yes':1,'No':0})
dataset['Obesity'] = dataset['Obesity'].map({'Yes':1,'No':0})
```

```
corrdata = dataset.corr()
```

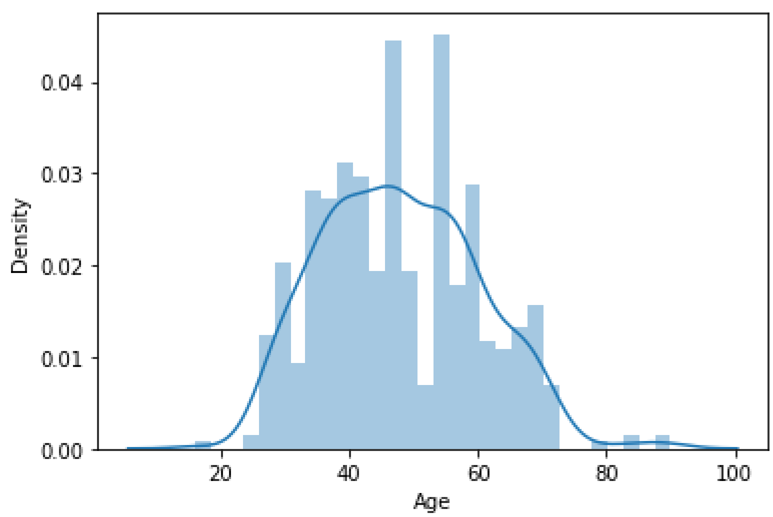
```
ax,fig·=·plt.subplots(figsize=(15,8))
sns.heatmap(corrdata,annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2d144a23d0>
```



```
sns.distplot(dataset['Age'],bins=30)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `d
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f2d13c7c0d0>
```




```
X1 = dataset.iloc[:,0:-1]
y1 = dataset.iloc[:,-1]
```

```
X1.columns
```

```
Index(['Age', 'Gender', 'Polyuria', 'Polydipsia', 'sudden weight loss',
       'weakness', 'Polyphagia', 'Genital thrush', 'visual blurring',
       'Itching', 'Irritability', 'delayed healing', 'partial paresis',
       'muscle stiffness', 'Alopecia', 'Obesity'],
      dtype='object')
```

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
best_feature = SelectKBest(score_func=chi2,k=10)
fit = best_feature.fit(X1,y1)
```

```
dataset_scores = pd.DataFrame(fit.scores_)
dataset_cols = pd.DataFrame(X1.columns)
```

```
featurescores = pd.concat([dataset_cols,dataset_scores],axis=1)
featurescores.columns=['column','scores']
featurescores
```

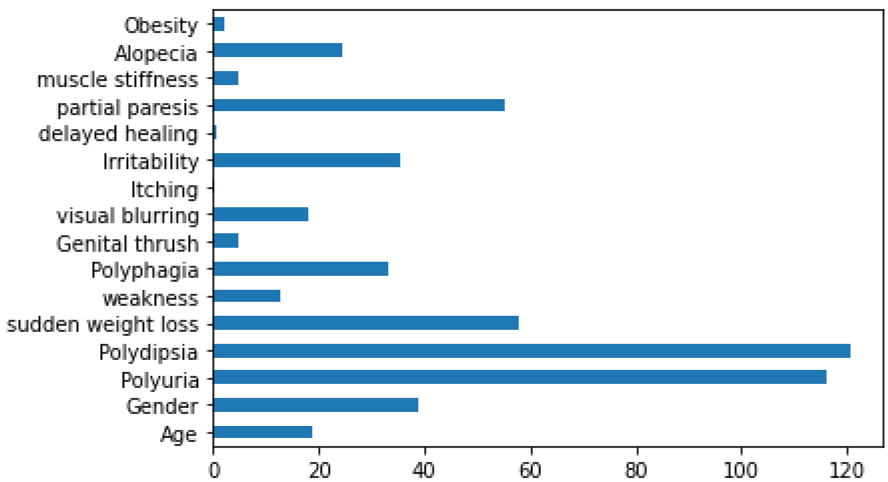
	column	scores
0	Age	18.845767
1	Gender	38.747637
2	Polyuria	116.184593
3	Polydipsia	120.785515
4	sudden weight loss	57.749309
5	weakness	12.724262
6	Polyphagia	33.198418
7	Genital thrush	4.914009
8	visual blurring	18.124571
9	Itching	0.047826
10	Irritability	35.334127
11	delayed healing	0.620188
12	partial paresis	55.314286
13	muscle stiffness	4.875000
14	Alopecia	24.402793
15	Obesity	2.250284

```
print(featurescores.nlargest(10,'scores'))
```

```
                column      scores
3           Polydipsia  120.785515
2             Polyuria  116.184593
4   sudden weight loss   57.749309
12     partial paresis   55.314286
1               Gender   38.747637
10        Irritability   35.334127
6           Polyphagia   33.198418
14            Alopecia   24.402793
0                  Age   18.845767
8      visual blurring   18.124571
```

```
featureview=pd.Series(fit.scores_, index=X1.columns)
featureview.plot(kind='barh')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2d0fd9e5d0>
```




```
from sklearn.feature_selection import VarianceThreshold
feature_high_variance = VarianceThreshold(threshold=(0.5*(1-0.5)))
falls=feature_high_variance.fit(X1)
```

```
dataset_scores1 = pd.DataFrame(falls.variances_)
dat1 = pd.DataFrame(X1.columns)
```

```
high_variance = pd.concat([dataset_scores1,dat1],axis=1)
high_variance.columns=['variance','cols']
high_variance[high_variance['variance']>0.2]
```

	variance	cols
0	147.374168	Age
1	0.232899	Gender
2	0.249985	Polyuria
3	0.247304	Polydipsia
4	0.243162	sudden weight loss
5	0.242511	weakness
6	0.248044	Polyphagia
8	0.247304	visual blurring
9	0.249819	Itching
11	0.248369	delayed healing

```
X = dataset[['Polydipsia','sudden weight loss','partial paresis','Irritability','Polyphagia',
y = dataset['class']
```

```
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=0)
```

```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
X_train = ss.fit_transform(X_train)
X_test = ss.transform(X_test)
```

Supervised Learning

```
from sklearn.svm import SVC
sv=SVC(kernel='linear',random_state=0)
sv.fit(X_train,y_train)
```

```
SVC(kernel='linear', random_state=0)
```

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator=sv, X=X_train ,y=y_train,cv=10)
print("accuracy is {:.2f} %".format(accuracies.mean()*100))
print("std is {:.2f} %".format(accuracies.std()*100))
```

```
accuracy is 83.18 %
std is 4.94 %
```

```
pre1=sv.predict(X_test)
```

```
svm_linear=accuracy_score(pre1,y_test)
print("Accuracy Score for SVM")
print(accuracy_score(pre1,y_test))
print("\nConfusion Matrix for SVM")
print(confusion_matrix(pre1,y_test))
```

```
Accuracy Score for SVM
0.9038461538461539

Confusion Matrix for SVM
[[34  4]
 [ 6 60]]
```

```
from·sklearn.metrics·import·classification_report
print(classification_report(pre1,y_test))
```

```
              precision    recall  f1-score   support

           0       0.85      0.89      0.87        38
           1       0.94      0.91      0.92        66

    accuracy                           0.90       104
   macro avg       0.89      0.90      0.90       104
weighted avg       0.91      0.90      0.90       104
```

```
from sklearn.svm import SVC
svrf=SVC(kernel='rbf',random_state=0)
svrf.fit(X_train,y_train)
```

```
SVC(random_state=0)
```

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator=svrf, X=X_train ,y=y_train,cv=10)
print("accuracy is {:.2f} %".format(accuracies.mean()*100))
print("std is {:.2f} %".format(accuracies.std()*100))
```

```
accuracy is 88.47 %
std is 3.69 %
```

```
pre2=svrf.predict(X_test)
```

```
svm_rbf=accuracy_score(pre2,y_test)
print("Accuracy·Score·for·SVM")
print(accuracy_score(pre2,y_test))
print("\nConfusion·Matrix·for·SVM")
print(confusion_matrix(pre2,y_test))
```

```
Accuracy Score for SVM
0.9807692307692307

Confusion Matrix for SVM
[[39  1]
 [ 1 63]]
```

```
from·sklearn.metrics·import·classification_report
print(classification_report(pre2,y_test))
```

```
              precision    recall  f1-score   support

           0       0.97      0.97      0.97        40
           1       0.98      0.98      0.98        64

    accuracy                           0.98       104
   macro avg       0.98      0.98      0.98       104
weighted avg       0.98      0.98      0.98       104
```

```
from sklearn.neighbors import KNeighborsClassifier
score=[]

for i in range(1,10):


    knn=KNeighborsClassifier(n_neighbors=i,metric='minkowski',p=2)
    knn.fit(X_train,y_train)
    pre3=knn.predict(X_test)
    ans=accuracy_score(pre3,y_test)
    score.append(round(100*ans,2))
print(sorted(score,reverse=True)[:5])
knn=sorted(score,reverse=True)[:1]
```

```
[98.08, 98.08, 98.08, 97.12, 96.15]
```

```
from sklearn.tree import DecisionTreeClassifier
dc=DecisionTreeClassifier(criterion='gini')
dc.fit(X_train,y_train)
```

```
DecisionTreeClassifier()
```

```
from·sklearn.model_selection·import·cross_val_score
accuracies·=·cross_val_score(estimator=dc,·X=X_train·,y=y_train,cv=10)
print("accuracy·is·{:.2f}·%".format(accuracies.mean()*100))
print("std·is·{:.2f}·%".format(accuracies.std()*100))
```

```
accuracy is 91.35 %
std is 3.91 %
```

```
pre5=dc.predict(X_test)


Decisiontress_classifier=accuracy_score(pre5,y_test)
print(accuracy_score(pre5,y_test))
print(confusion_matrix(pre5,y_test))
```

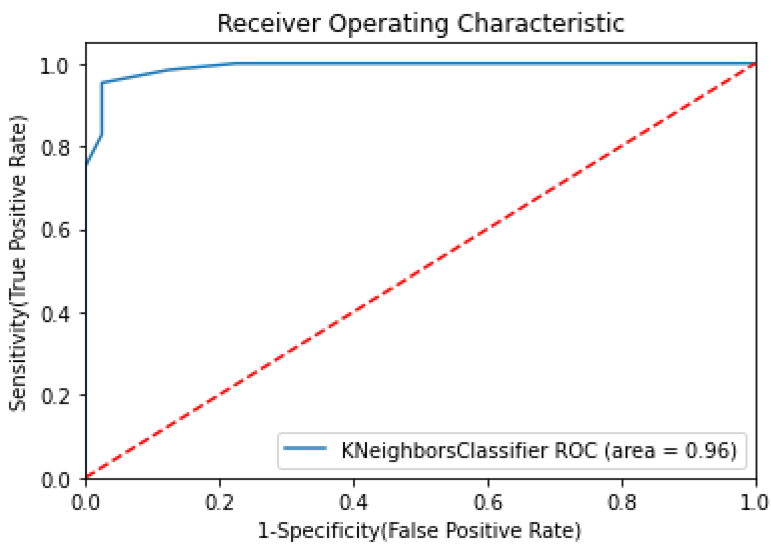
```
0.9711538461538461
[[39  2]
 [ 1 62]]
```

```
from sklearn.metrics import classification_report
print(classification_report(pre5,y_test))
```

```
              precision    recall  f1-score   support

           0       0.97      0.95      0.96        41
           1       0.97      0.98      0.98        63

    accuracy                           0.97       104
   macro avg       0.97      0.97      0.97       104
weighted avg       0.97      0.97      0.97       104
```

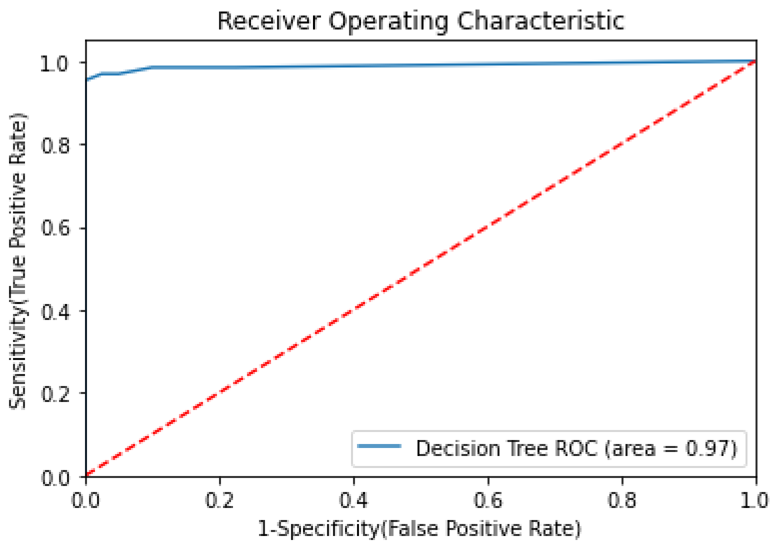
```
from sklearn import metrics
import matplotlib.pyplot as plt

plt.figure()

models = [


    {
    'label': 'KNeighborsClassifier',
    'model': KNeighborsClassifier(n_neighbors=i,metric='minkowski',p=2),
},


]

for m in models:
    model = m['model']
    model.fit(X_train, y_train)
    y_pred=model.predict(X_test)

fpr, tpr, thresholds = metrics.roc_curve(y_test, model.predict_proba(X_test)[:,1])

auc = metrics.roc_auc_score(y_test,model.predict(X_test))
```

```
plt.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % (m['label'], auc))

plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('1-Specificity(False Positive Rate)')
plt.ylabel('Sensitivity(True Positive Rate)')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```




```
from sklearn import metrics
import matplotlib.pyplot as plt

plt.figure()

models = [

{
    'label': 'Decision Tree',
    'model': DecisionTreeClassifier(criterion='gini'),
},

]

for m in models:
    model = m['model']
    model.fit(X_train, y_train)
    y_pred=model.predict(X_test)

fpr, tpr, thresholds = metrics.roc_curve(y_test, model.predict_proba(X_test)[:,1])

auc = metrics.roc_auc_score(y_test,model.predict(X_test))

plt.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % (m['label'], auc))
```

```
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('1-Specificity(False Positive Rate)')
plt.ylabel('Sensitivity(True Positive Rate)')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```



Unsupervised Learning

```
from sklearn.cluster import KMeans
KMeans_Clustering = KMeans(n_clusters =2, random_state=0)
KMeans_Clustering.fit(X_train)
```

```
KMeans(n_clusters=2, random_state=0)
```

```
kpred = KMeans_Clustering.predict(X_test)
kpred
```

```
array([1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1,
       1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1,
       0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0,
       0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
       0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0], dtype=int32)
```

```
print(accuracy_score(kpred,y_test))
```

```
0.7403846153846154
```

```
import sklearn
print('Classification report:\n\n', sklearn.metrics.classification_report(y_test,kpred))
```

```
Classification report:

               precision    recall  f1-score   support

           0       0.62      0.82      0.71        40
           1       0.86      0.69      0.77        64

    accuracy                           0.74       104
   macro avg       0.74      0.76      0.74       104
weighted avg       0.77      0.74      0.74       104
```

```
from sklearn.metrics import confusion_matrix
print("Confusion Matrix :")

confusion_matrix(y_test, kpred),
```

```
Confusion Matrix :
(array([[33,  7],
        [20, 44]]),)
```

Deep Learning

```
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(11,11,11), max_iter=1000)
mlp.fit(X_train, y_train.values.ravel())
```

```
MLPClassifier(hidden_layer_sizes=(11, 11, 11), max_iter=1000)
```

```
predictions = mlp.predict(X_test)
print(predictions)
```

```
[1 1 1 0 0 1 1 1 0 1 1 0 1 1 1 1 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 1 1 1 1
 0 1 1 1 0 1 0 0 1 1 0 1 0 1 0 1 0 1 1 1 1 1 1 0 0 1 0 1 1 1 0 1 1 0 0 1 1
 0 1 0 1 1 0 1 1 0 0 1 0 1 1 0 0 0 0 0 0 0 1 1 1 1 0 0 1 1 1]
```

```
print('Accuracy: {:.2f}'.format(accuracy_score(y_test, predictions)))
```

```
Accuracy: 0.97
```

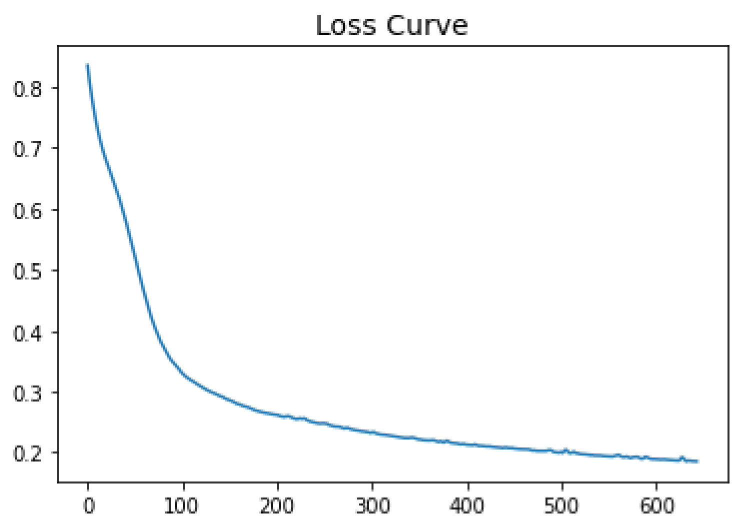
```
print(confusion_matrix(y_test,predictions))
```

```
[[39  1]
 [ 1 63]]
```

```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.97      0.97      0.97        40
           1       0.98      0.98      0.98        64

    accuracy                           0.98       104
   macro avg       0.98      0.98      0.98       104
weighted avg       0.98      0.98      0.98       104
```

```
plt.plot(mlp.loss_curve_)
plt.title("Loss Curve", fontsize=14)
plt.show()
```




```
hyper_params = {
    'task': 'train',
    'boosting_type': 'gbdt',
    'objective': 'regression',
    'metric': ['l1','l2'],
    'learning_rate': 0.005,
    'feature_fraction': 0.7,
    'bagging_fraction': 0.6,
    'bagging_freq': 10,
    'verbose': 0,
    "max_depth": 12,
#     "num_leaves": 128,
#     "max_bin": 512,
    "num_iterations": 10000
}
```

```
from lightgbm import *


gbm = LGBMRegressor(**hyper_params)


gbm_model = gbm.fit(X_train, y_train,
        eval_set=[(X_test, y_test)],
        eval_metric='RMSE',
        early_stopping_rounds=100)
```

```
[2310]  valid_0's l1: 0.17313   valid_0's rmse: 0.237158        valid_0's l2: 0.0562
[2311]  valid_0's l1: 0.173168  valid_0's rmse: 0.237143        valid_0's l2: 0.0562
[2312]  valid_0's l1: 0.173232  valid_0's rmse: 0.237156        valid_0's l2: 0.0562
[2313]  valid_0's l1: 0.173222  valid_0's rmse: 0.237122        valid_0's l2: 0.0562
[2314]  valid_0's l1: 0.173287  valid_0's rmse: 0.237137        valid_0's l2: 0.0562
[2315]  valid_0's l1: 0.173287  valid_0's rmse: 0.237132        valid_0's l2: 0.0562
[2316]  valid_0's l1: 0.173325  valid_0's rmse: 0.237118        valid_0's l2: 0.0562
[2317]  valid_0's l1: 0.173329  valid_0's rmse: 0.237117        valid_0's l2: 0.0562
[2318]  valid_0's l1: 0.173394  valid_0's rmse: 0.237132        valid_0's l2: 0.0562
[2319]  valid_0's l1: 0.173431  valid_0's rmse: 0.237119        valid_0's l2: 0.0562
[2320]  valid_0's l1: 0.173468  valid_0's rmse: 0.237106        valid_0's l2: 0.0562
[2321]  valid_0's l1: 0.173511  valid_0's rmse: 0.237092        valid_0's l2: 0.0562
[2322]  valid_0's l1: 0.173539  valid_0's rmse: 0.237065        valid_0's l2: 0.0562
[2323]  valid_0's l1: 0.173589  valid_0's rmse: 0.237058        valid_0's l2: 0.0561
[2324]  valid_0's l1: 0.173615  valid_0's rmse: 0.237035        valid_0's l2: 0.0561
[2325]  valid_0's l1: 0.173627  valid_0's rmse: 0.237028        valid_0's l2: 0.0561
[2326]  valid_0's l1: 0.173664  valid_0's rmse: 0.237037        valid_0's l2: 0.0561
[2327]  valid_0's l1: 0.173689  valid_0's rmse: 0.237014        valid_0's l2: 0.0561
[2328]  valid_0's l1: 0.173715  valid_0's rmse: 0.237007        valid_0's l2: 0.0561
[2329]  valid_0's l1: 0.173744  valid_0's rmse: 0.237021        valid_0's l2: 0.0561
[2330]  valid_0's l1: 0.173786  valid_0's rmse: 0.237008        valid_0's l2: 0.0561
[2331]  valid_0's l1: 0.173805  valid_0's rmse: 0.237005        valid_0's l2: 0.0561
[2332]  valid_0's l1: 0.173806  valid_0's rmse: 0.236998        valid_0's l2: 0.0561
[2333]  valid_0's l1: 0.173807  valid_0's rmse: 0.236984        valid_0's l2: 0.0561
[2334]  valid_0's l1: 0.173747  valid_0's rmse: 0.236919        valid_0's l2: 0.0561
[2335]  valid_0's l1: 0.173756  valid_0's rmse: 0.236919        valid_0's l2: 0.0561
[2336]  valid_0's l1: 0.173779  valid_0's rmse: 0.236918        valid_0's l2: 0.0561
[2337]  valid_0's l1: 0.173781  valid_0's rmse: 0.236916        valid_0's l2: 0.0561
[2338]  valid_0's l1: 0.173784  valid_0's rmse: 0.236901        valid_0's l2: 0.0561
[2339]  valid_0's l1: 0.173779  valid_0's rmse: 0.236876        valid_0's l2: 0.0561
[2340]  valid_0's l1: 0.173743  valid_0's rmse: 0.236834        valid_0's l2: 0.0560
[2341]  valid_0's l1: 0.17371   valid_0's rmse: 0.236819        valid_0's l2: 0.0560
[2342]  valid_0's l1: 0.173688  valid_0's rmse: 0.236803        valid_0's l2: 0.0560
[2343]  valid_0's l1: 0.173676  valid_0's rmse: 0.236798        valid_0's l2: 0.0560
[2344]  valid_0's l1: 0.173656  valid_0's rmse: 0.236787        valid_0's l2: 0.0560
[2345]  valid_0's l1: 0.173632  valid_0's rmse: 0.236779        valid_0's l2: 0.0560
[2346]  valid_0's l1: 0.173627  valid_0's rmse: 0.236752        valid_0's l2: 0.0560
[2347]  valid_0's l1: 0.173617  valid_0's rmse: 0.236737        valid_0's l2: 0.0560
[2348]  valid_0's l1: 0.173609  valid_0's rmse: 0.236715        valid_0's l2: 0.0560
[2349]  valid_0's l1: 0.173589  valid_0's rmse: 0.2367  valid_0's l2: 0.0560269
[2350]  valid_0's l1: 0.17358   valid_0's rmse: 0.236678        valid_0's l2: 0.0560
[2351]  valid_0's l1: 0.173551  valid_0's rmse: 0.236674        valid_0's l2: 0.0560
[2352]  valid_0's l1: 0.173524  valid_0's rmse: 0.23665 valid_0's l2: 0.0560033
[2353]  valid_0's l1: 0.173488  valid_0's rmse: 0.23662 valid_0's l2: 0.0559891
[2354]  valid_0's l1: 0.173451  valid_0's rmse: 0.236594        valid_0's l2: 0.0559
```

```
[2355]  valid_0's l1: 0.17342   valid_0's rmse: 0.236574       valid_0's l2: 0.0559
[2356]  valid_0's l1: 0.173388  valid_0's rmse: 0.236556       valid_0's l2: 0.0559
[2357]  valid_0's l1: 0.173352  valid_0's rmse: 0.236544       valid_0's l2: 0.0559
[2358]  valid_0's l1: 0.173315  valid_0's rmse: 0.23654 valid_0's l2: 0.0559512
[2359]  valid_0's l1: 0.173305  valid_0's rmse: 0.236521       valid_0's l2: 0.0559
[2360]  valid_0's l1: 0.173261  valid_0's rmse: 0.236492       valid_0's l2: 0.0559
[2361]  valid_0's l1: 0.173234  valid_0's rmse: 0.236487       valid_0's l2: 0.0559
[2362]  valid_0's l1: 0.173215  valid_0's rmse: 0.236468       valid_0's l2: 0.0559
[2363]  valid_0's l1: 0.173203  valid_0's rmse: 0.236458       valid_0's l2: 0.0559
Early stopping, best iteration is:
[2263]  valid_0's l1: 0.172941  valid_0's rmse: 0.237576       valid_0's l2: 0.0564
```

```
pred_tar = gbm_model.predict(X_test[:1000000])
```

```
pred_tar
```

```
array([ 9.01898158e-01,  8.67658853e-01,  1.03822187e+00,  7.35419891e-02,
        5.05479774e-01,  8.46197022e-01,  9.45429424e-01,  7.57195030e-01,
        3.41959739e-01,  1.08241770e+00,  9.41112066e-01, -3.60251492e-02,
        7.70245597e-01,  6.25232812e-01,  8.12002207e-01,  9.83131168e-01,
        3.60858318e-01,  1.09232010e+00,  7.23540854e-01,  9.83131168e-01,
        8.84374124e-01,  7.15908815e-01,  9.95469619e-01,  1.40736274e-01,
        1.04963382e-01,  3.41959739e-01,  8.87028801e-01,  7.64060331e-01,
        8.51277379e-01,  8.78123154e-01,  1.07201572e+00,  2.74855492e-01,
        3.28859949e-01,  6.25232812e-01,  8.65816979e-01,  8.12461387e-01,
        1.14390427e+00,  3.27527956e-01,  1.03822187e+00,  9.17252925e-01,
        9.33135311e-01,  5.12414829e-02,  1.05102897e+00,  3.76022525e-01,
        1.04963382e-01,  9.83131168e-01,  9.57293321e-01,  2.82061931e-01,
        9.57293321e-01,  3.56060194e-01,  7.33091615e-01,  3.41959739e-01,
        9.79608157e-01,  7.35419891e-02,  9.57293321e-01,  4.75657783e-01,
        1.00069429e+00,  8.87286325e-01,  9.82048568e-01,  9.83131168e-01,
        2.83479012e-01,  1.08302698e-01,  1.08241770e+00,  6.01830807e-01,
        9.45076206e-01,  4.25186486e-01,  8.99756839e-01,  3.76022525e-01,
        1.00668187e+00,  9.57293321e-01, -3.67934338e-02,  3.13632573e-01,
        9.19177362e-01,  1.04818521e+00,  7.35419891e-02,  9.01898158e-01,
       -7.07243665e-02,  7.05993427e-01,  1.02784072e+00,  7.35419891e-02,
        8.60200383e-01,  1.09409055e+00, -3.60251492e-02,  7.35419891e-02,
        8.51277379e-01,  1.79596089e-01,  9.44245482e-01,  8.84374124e-01,
        8.40252140e-02,  8.27773617e-03,  8.44977406e-02,  8.40252140e-02,
        9.67450537e-04,  3.27527956e-01,  5.05479774e-01,  9.17696166e-01,
        8.78136545e-01,  6.40923775e-01,  4.45994095e-01,  7.35419891e-02,
        3.13632573e-01,  8.55568152e-01,  1.21274647e+00,  7.90056548e-01])
```

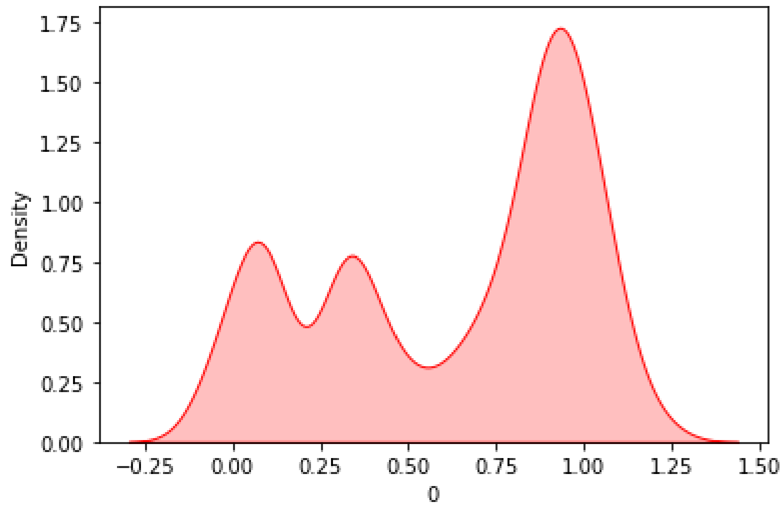
```
print("\nTest  R2 Score : %.2f"%gbm.score(X_train, y_train))
print("Train R2 Score : %.2f"%gbm.score(X_test, y_test))
```

```
Test  R2 Score : 0.73
Train R2 Score : 0.76
```

```
import seaborn as sns
```

```
sns.kdeplot(pred_tar[0], shade=True, bw=0.2, color="red")
plt.figure(figsize=(20,10))
plt.show()
```

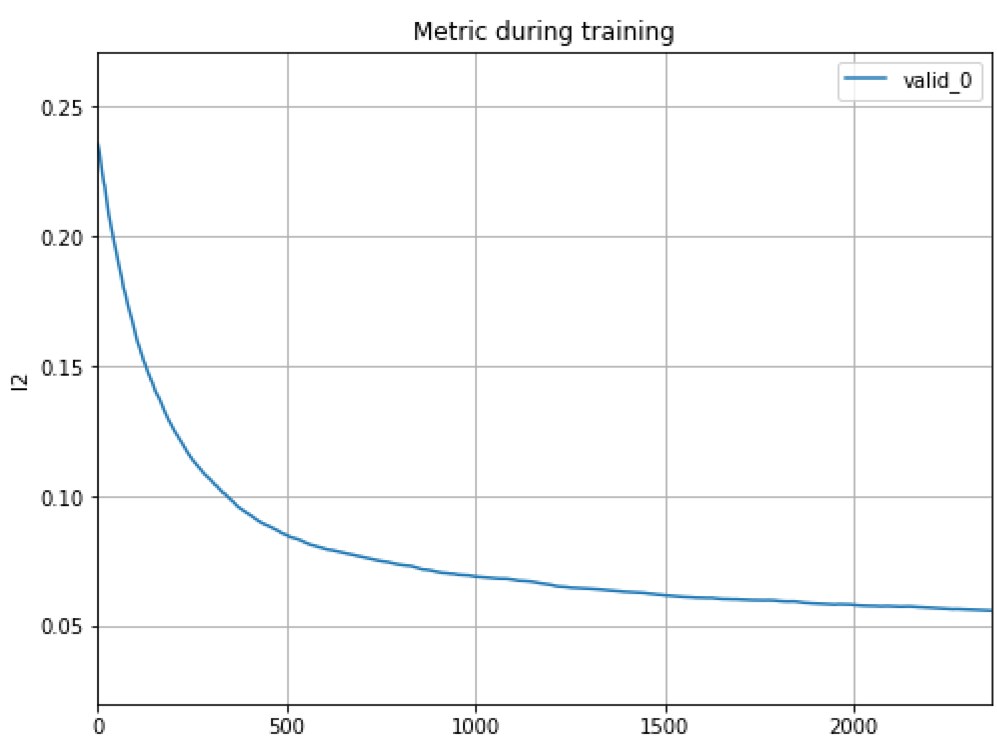
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:1699: FutureWarning: Th
  warnings.warn(msg, FutureWarning)
```



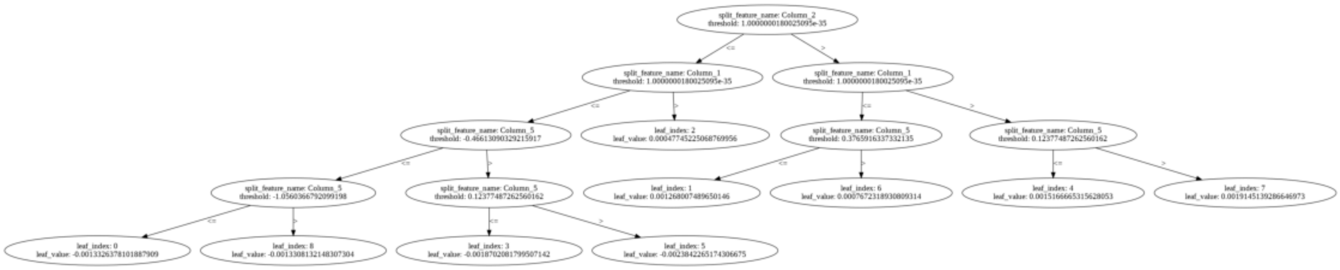

```
<Figure size 1440x720 with 0 Axes>
```

```
import lightgbm as lgb
lgb.plot_metric(gbm, figsize=(8,6));
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: UserWarning: more than
```




```
lgb.plot_tree(gbm, tree_index = 1, figsize=(20,15));
```



✓ 0s completed at 3:23 PM