Exploratory Data Analysis (EDA)

In [2]:
```
#Load Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [3]:
```
# Load the dataset
dataset = pd.read_csv('wireless_churn.csv')
dataset.head()
```

Out[3]:

	AccountWeeks	ContractRenewal	DataPlan	DataUsage	CustServCalls	DayMins	DayCalls	MonthlyCharge	OverageFee	RoamMins	Churn
0	128	1	1	2.7	1	265.1	110	89.0	9.87	10.0	0
1	107	1	1	3.7	1	161.6	123	82.0	9.78	13.7	0
2	137	1	0	0.0	0	243.4	114	52.0	6.06	12.2	0
3	84	0	0	0.0	2	299.4	71	57.0	3.10	6.6	0
4	75	0	0	0.0	3	166.7	113	41.0	7.42	10.1	0

In [4]:
```
dataset.describe()
```

Out[4]:

	AccountWeeks	ContractRenewal	DataPlan	DataUsage	CustServCalls	DayMins	DayCalls	MonthlyCharge	OverageFee	F
count	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	3333.000000	33
mean	101.064806	0.903090	0.276628	0.816475	1.562856	179.775098	100.435644	56.305161	10.051488	
std	39.822106	0.295879	0.447398	1.272668	1.315491	54.467389	20.069084	16.426032	2.535712	
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	14.000000	0.000000	
25%	74.000000	1.000000	0.000000	0.000000	1.000000	143.700000	87.000000	45.000000	8.330000	
50%	101.000000	1.000000	0.000000	0.000000	1.000000	179.400000	101.000000	53.500000	10.070000	
75%	127.000000	1.000000	1.000000	1.780000	2.000000	216.400000	114.000000	66.200000	11.770000	
max	243.000000	1.000000	1.000000	5.400000	9.000000	350.800000	165.000000	111.300000	18.190000	

In [5]:
```
from pandas_profiling import ProfileReport
# Conduct EDA
profile = ProfileReport(dataset, title='Wireless Churn Dataset Report')
profile.to_file('Wireless_Churn_EDA_Report.html')
```

```
C:\Users\PAVILION\AppData\Local\Temp\ipykernel_20592\4244276527.py:1: DeprecationWarning: `import pandas_profil
ing` is going to be deprecated by April 1st. Please use `import ydata_profiling` instead.
  from pandas_profiling import ProfileReport
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]
Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]
Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]
Export report to file:   0%|          | 0/1 [00:00<?, ?it/s]
```

In [6]:
```
#Define x and y variable
x = dataset.drop('Churn',axis=1).to_numpy()
y = dataset['Churn'].to_numpy()

# Create Train and Test Datasets
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20,stratify=y,random_state=100)
```

Remove Anomalies

In [7]:
```
# Use built-in isolation forest
from sklearn.ensemble import IsolationForest

# The prediction returns 1 if sample point is inlier. If outlier prediction returns -1
clf_all_features = IsolationForest(random_state=100)
clf_all_features.fit(x_train)

#Predict if a particular sample is an outlier using all features for higher dimensional data set.
y_pred_train = clf_all_features.predict(x_train)
y_pred_train2 =np.array(list(map(lambda x: x == 1, y_pred_train)))

# Exclude suggested outlier samples for improvement of prediction power/score
x_train_mod = x_train[y_pred_train2, ]
y_train_mod = y_train[y_pred_train2, ]

#Size of Datasets
print('Original Train Dataset Size : {}'.format(len(x_train)))
```

```
print('New Train Dataset Size      : {}'.format(len(x_train_mod)))
```

```
Original Train Dataset Size : 2666
New Train Dataset Size      : 2124
```

Create Learning Curves

In [8]:

```
#Scale the Data
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train2 = sc.fit_transform(x_train)
x_test2 = sc.transform(x_test)

#Model
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
```

In [9]:

```
#Base Logistical Regression Model
from sklearn.metrics import classification_report, confusion_matrix

for name,method in [('LogReg', LogisticRegression(solver='lbfgs',class_weight='balanced',max_iter=1000,
                                                  random_state=100))]:
    method.fit(x_train2,y_train)
    predict = method.predict(x_test2)
    print('\nEstimator: {}'.format(name))
    print(confusion_matrix(y_test,predict))
    print(classification_report(y_test,predict))
```

```
Estimator: LogReg
[[430 140]
 [ 24  73]]
              precision    recall  f1-score   support

           0       0.95      0.75      0.84       570
           1       0.34      0.75      0.47        97

    accuracy                           0.75       667
   macro avg       0.64      0.75      0.66       667
weighted avg       0.86      0.75      0.79       667
```

In [10]:

```
#Construct some pipelines
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

#Create Pipeline

pipeline =[]

pipe_logreg = Pipeline([('scl', StandardScaler()),
                    ('clf', LogisticRegression(solver='lbfgs',class_weight='balanced',max_iter=1000,
                                               random_state=100))])
pipeline.insert(0,pipe_logreg)

pipe_gnb = Pipeline([('scl', StandardScaler()),
                    ('clf', GaussianNB())])

pipeline.insert(1,pipe_gnb)

#Set grid search params

modelpara =[]

param_gridlogreg = {'clf__C': [0.01, 0.1, 1, 10, 100],
                    'clf__penalty': ['l2']}
modelpara.insert(0,param_gridlogreg)
```

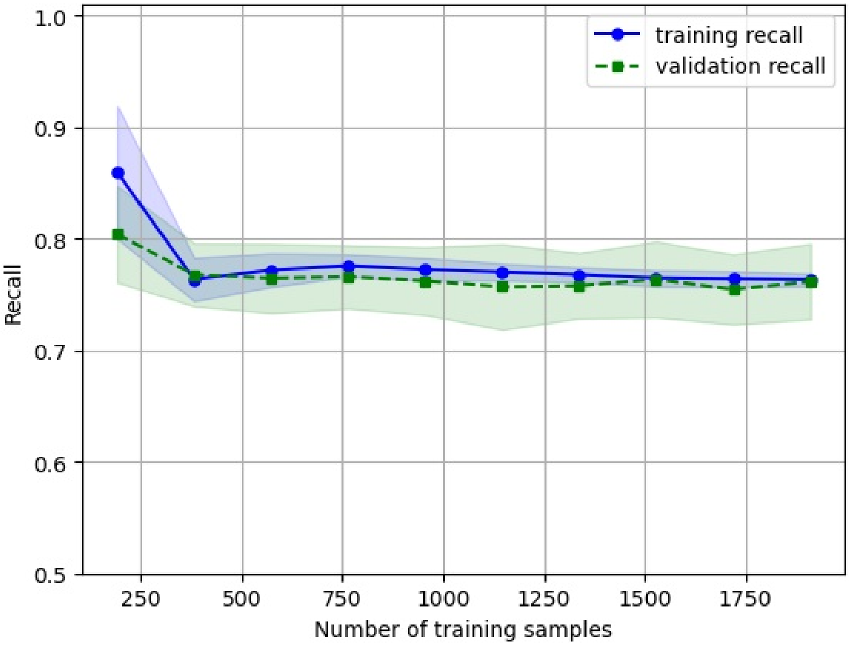
In [12]:

```
#Define Plot for learning curve

from sklearn.model_selection import learning_curve

def plot_learning_curves(model):
    train_sizes, train_scores, test_scores = learning_curve(estimator=model,
                                                            X=x_train_mod,
                                                            y=y_train_mod,
                                                            train_sizes= np.linspace(0.1, 1.0, 10),
                                                            cv=10,
                                                            scoring='recall_weighted',random_state=100)
    train_mean = np.mean(train_scores, axis=1)
    train_std = np.std(train_scores, axis=1)
    test_mean = np.mean(test_scores, axis=1)
    test_std = np.std(test_scores, axis=1)

    plt.plot(train_sizes, train_mean,color='blue', marker='o',
             markersize=5, label='training recall')
    plt.fill_between(train_sizes, train_mean + train_std, train_mean - train_std,
                     alpha=0.15, color='blue')
```

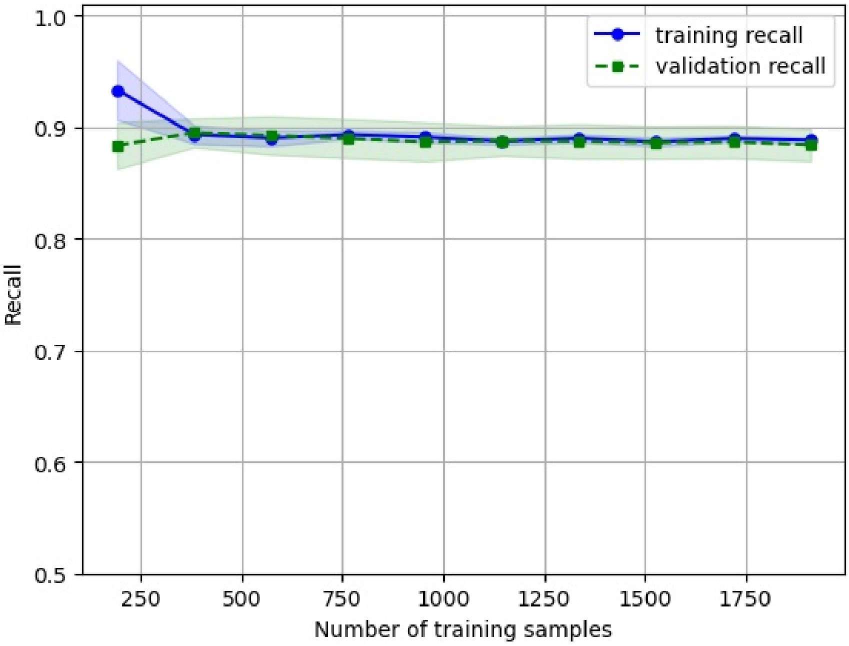
```
    plt.plot(train_sizes, test_mean, color='green', linestyle='--', marker='s', markersize=5,
             label='validation recall')
    plt.fill_between(train_sizes, test_mean + test_std, test_mean - test_std,
                     alpha=0.15, color='green')
    plt.grid(True)
    plt.xlabel('Number of training samples')
    plt.ylabel('Recall')
    plt.legend(loc='best')
    plt.ylim([0.5, 1.01])
    plt.show()
```

In [13]:
```
#Plot Learning Curve
print('Logisistic Regression - Learning Curve')
plot_learning_curves(pipe_logreg)
print('GNB Learning Curve')
plot_learning_curves(pipe_gnb)
```

```
Logisistic Regression - Learning Curve
```




```
GNB Learning Curve
```



Optimize Models

In [14]:
```
#Prepare Models
from sklearn import model_selection
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

In [15]:
```
#Model Analysis
from sklearn.model_selection import RepeatedKFold
from sklearn.model_selection import cross_val_score
```

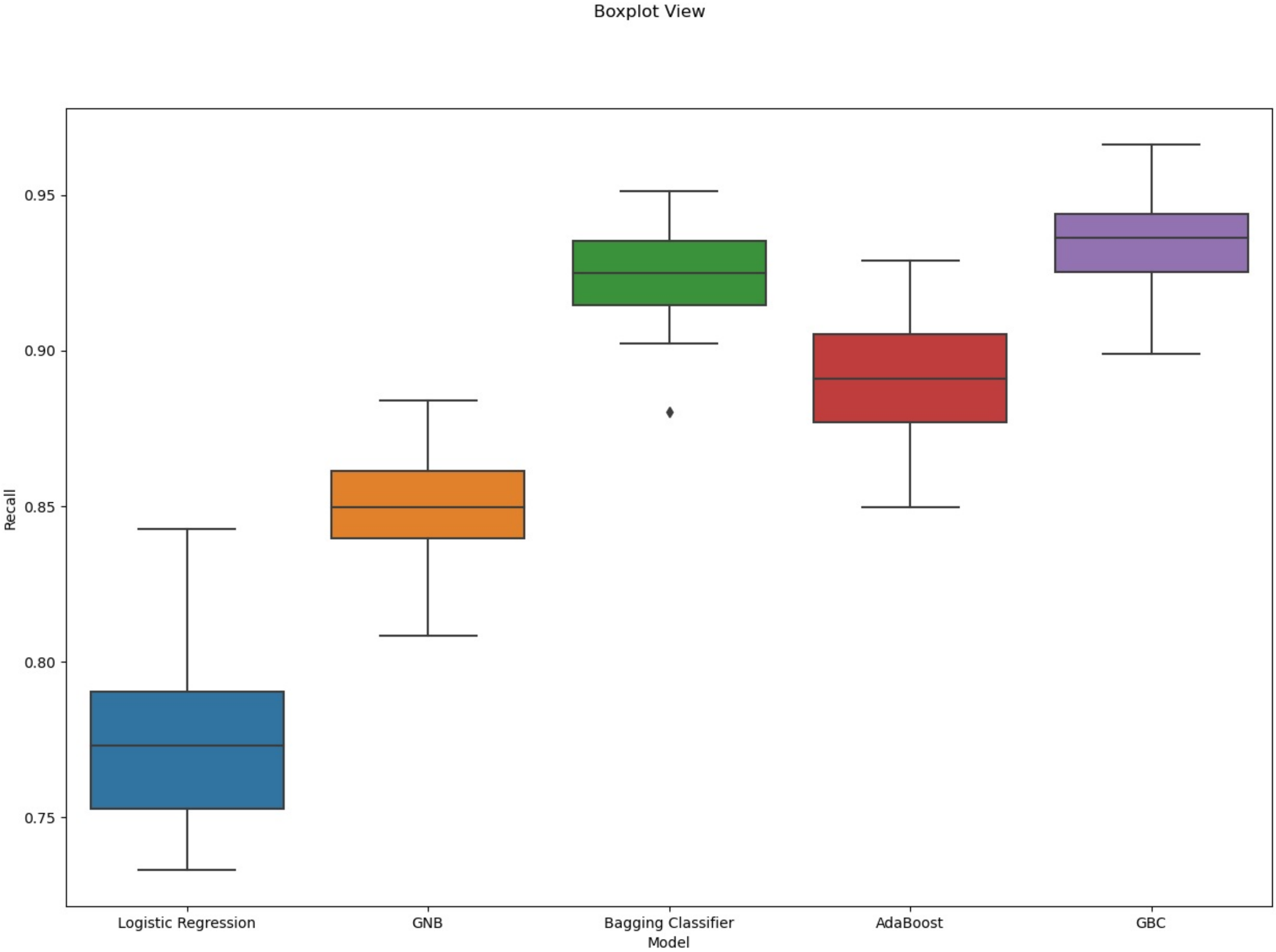
```python
modelAnalysis=[]
modelAnalysis.append(('Logistic Regression',LogisticRegression(solver='lbfgs',class_weight='balanced',
                                                        max_iter=1000,random_state=100)))
modelAnalysis.append(('GNB', GaussianNB()))
modelAnalysis.append(('Bagging Classifier',BaggingClassifier(random_state=100)))
modelAnalysis.append(('AdaBoost',AdaBoostClassifier(random_state=100)))
modelAnalysis.append(('GBC',GradientBoostingClassifier(random_state=100)))


#Model Evaluation
results =[]
names=[]
scoring ='recall_weighted'
print('Model Evaluation - Recall Score')
for name, model in modelAnalysis:
    rkf=RepeatedKFold(n_splits=10, n_repeats=5, random_state=100)
    cv_results = cross_val_score(model,x_train2,y_train,cv=rkf,scoring=scoring)
    results.append(cv_results)
    names.append(name)
    print('{} {:.2f} +/- {:.2f}'.format(name,cv_results.mean(),cv_results.std()))
print('\n')

#Boxpot View
fig = plt.figure(figsize=(15,10))
fig.suptitle('Boxplot View')
ax = fig.add_subplot(111)
sns.boxplot(data=results)
ax.set_xticklabels(names)
plt.ylabel('Recall')
plt.xlabel('Model')
plt.show()
```

```
Model Evaluation - Recall Score
Logistic Regression 0.77 +/- 0.02
GNB 0.85 +/- 0.02
Bagging Classifier 0.92 +/- 0.01
```

```
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
```

```
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent
this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The
```

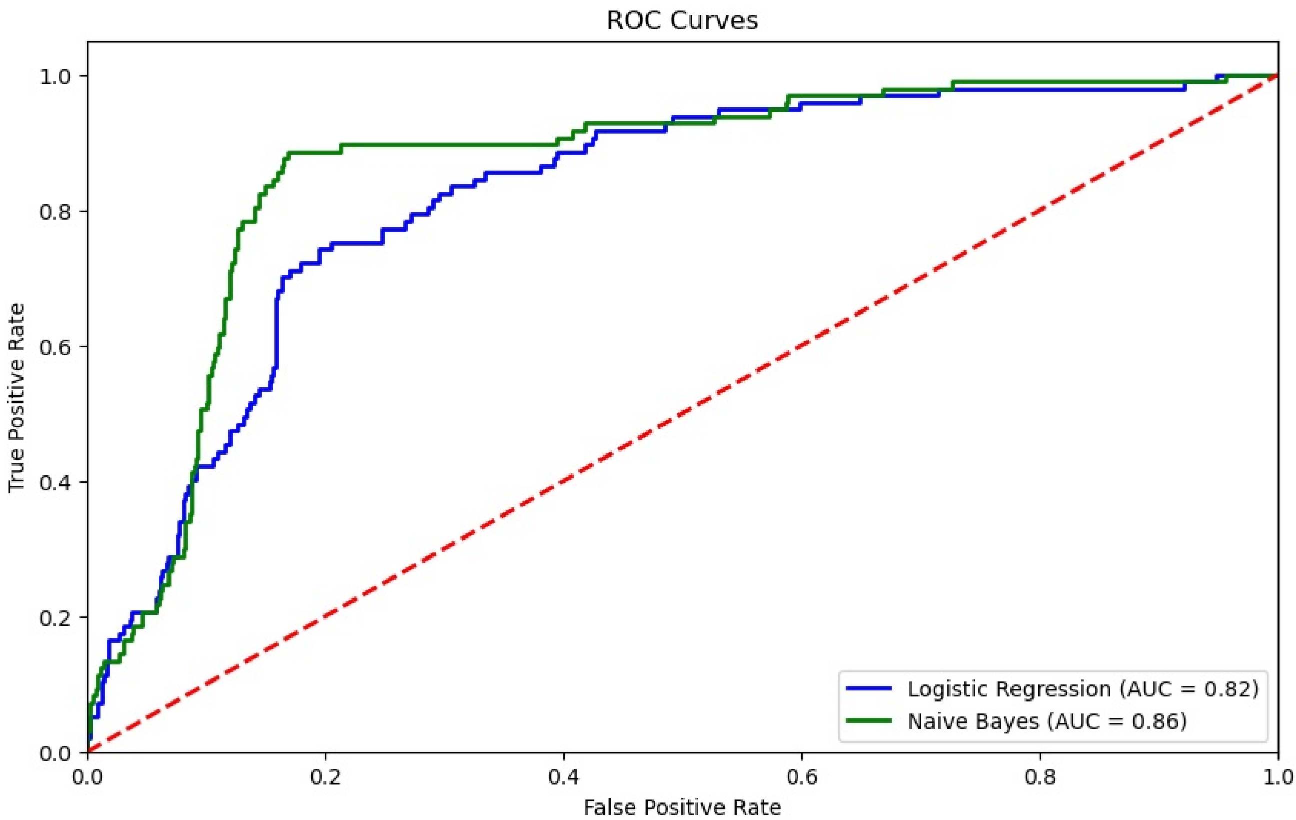
```
SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\PAVILION\anaconda3\envs\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:527: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
AdaBoost 0.89 +/- 0.02
GBC 0.93 +/- 0.01
```



In [16]:
```
from sklearn.metrics import roc_auc_score, roc_curve, auc
```

In [22]:
```
# Logistic Regression
from sklearn.metrics import classification_report, confusion_matrix
log_reg = LogisticRegression(solver='lbfgs', class_weight='balanced', max_iter=1000, random_state=100)
log_reg.fit(x_train2, y_train)
y_pred_prob_log_reg = log_reg.predict_proba(x_test2)[:, 1]

# Naive Bayes
nb = GaussianNB()
nb.fit(x_train2, y_train)
y_pred_prob_nb = nb.predict_proba(x_test2)[:, 1]

# Calculate ROC AUC scores
roc_auc_log_reg = roc_auc_score(y_test, y_pred_prob_log_reg)
roc_auc_nb = roc_auc_score(y_test, y_pred_prob_nb)

# Calculate ROC curves
fpr_log_reg, tpr_log_reg, _ = roc_curve(y_test, y_pred_prob_log_reg)
fpr_nb, tpr_nb, _ = roc_curve(y_test, y_pred_prob_nb)
```

```
# Plot ROC curves
plt.figure(figsize=(10, 6))
plt.plot(fpr_log_reg, tpr_log_reg, color='blue', lw=2, label=f'Logistic Regression (AUC = {roc_auc_log_reg:.2f}
plt.plot(fpr_nb, tpr_nb, color='green', lw=2, label=f'Naive Bayes (AUC = {roc_auc_nb:.2f})')
plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curves')
plt.legend(loc='lower right')
plt.show()
method.fit(x_train2,y_train)
predict= method.predict(x_test2)
print('\nEstimator: {}'.format(name))
print(confusion_matrix(y_test,predict))
print(classification_report(y_test,predict))
```




```
Estimator: GBC
[[430 140]
 [ 24  73]]
              precision    recall  f1-score   support

           0       0.95      0.75      0.84       570
           1       0.34      0.75      0.47        97

    accuracy                           0.75       667
   macro avg       0.64      0.75      0.66       667
weighted avg       0.86      0.75      0.79       667
```

5 Ensemble Voting Model

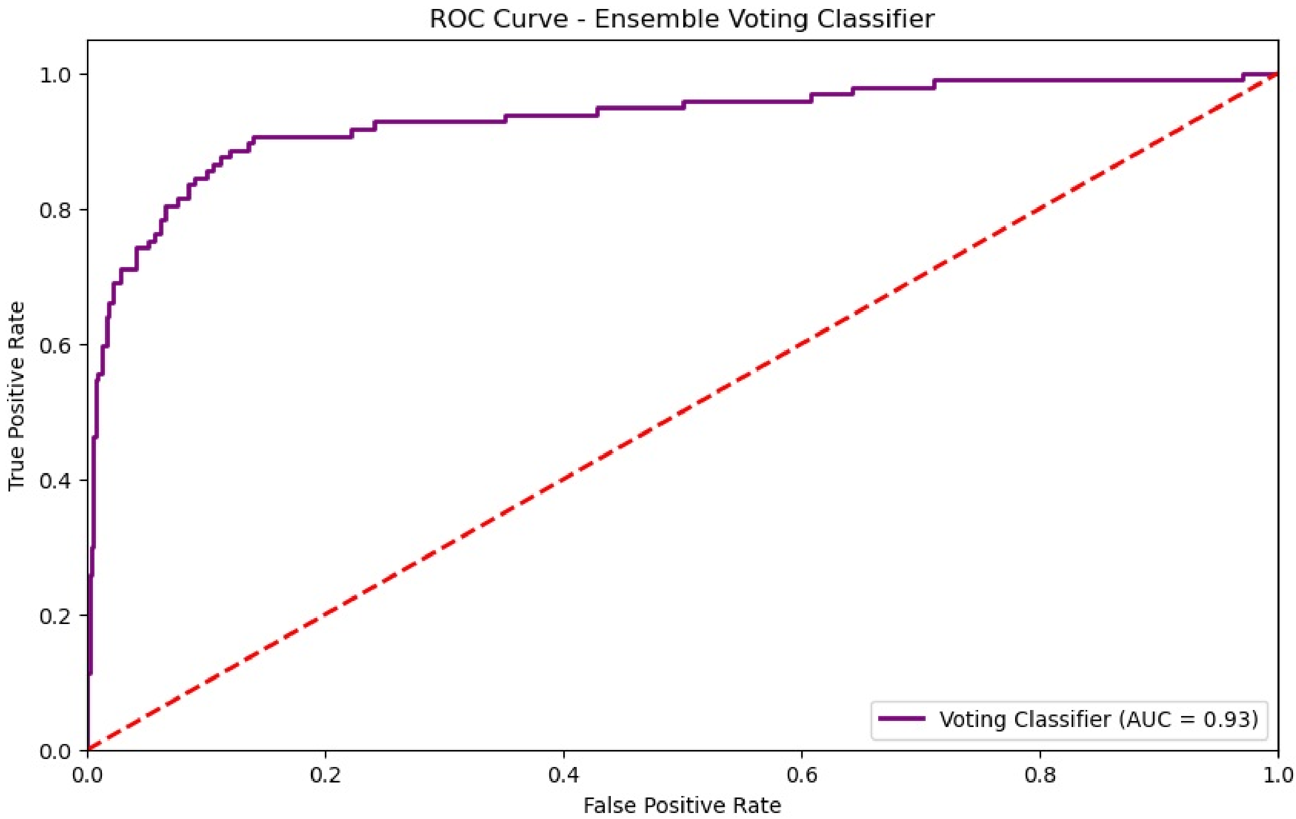
In [23]:

```
from sklearn.ensemble import GradientBoostingClassifier, VotingClassifier
```

In [24]:

```
# Logistic Regression
log_reg = LogisticRegression(solver='lbfgs', class_weight='balanced', max_iter=1000, random_state=100)

# Gradient Boosting
gb = GradientBoostingClassifier(n_estimators=100, random_state=100)

# Voting Classifier
voting_clf = VotingClassifier(estimators=[('lr', log_reg), ('gb', gb)], voting='soft')

# Train the voting classifier
voting_clf.fit(x_train2, y_train)
y_pred_prob_voting = voting_clf.predict_proba(x_test2)[:, 1]

# Calculate ROC AUC score
roc_auc_voting = roc_auc_score(y_test, y_pred_prob_voting)

# Calculate ROC curve
```

```
fpr_voting, tpr_voting, _ = roc_curve(y_test, y_pred_prob_voting)

# Plot ROC curve
plt.figure(figsize=(10, 6))
plt.plot(fpr_voting, tpr_voting, color='purple', lw=2, label=f'Voting Classifier (AUC = {roc_auc_voting:.2f})')
plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Ensemble Voting Classifier')
plt.legend(loc='lower right')
plt.show()
```



In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js