Misr University for Science & Technology
Faculty of Information Technology
Artificial Intelligence Dept.
AI443: Machine Learning
Fall 2023-2024

AI443 Final Project Documentation

Smoke Detection

Submitted by:

No.	ID	Name	Section	Mark/5
1	94071	Mostafa Tarek	3 - 5	
2	94303	Mohamed Ezz	3 - 5	
3	94218	Abdelrahman Ahmed	3 - 5	

Supervised by:

1- Assoc.Prof/ Mohand Daef
2- Eng/ Shaimaa Bahaa

Code /10	Documentation /5

Problem Statement:

The timely and accurate detection of smoke is critical for public safety and disaster prevention. Conventional smoke detection methods often lack speed and effectiveness, especially in diverse environmental conditions. To overcome these limitations, there is a need to explore advanced technologies at the intersection of machine learning and computer vision.

This research focuses on the development of robust Smoke Detection Systems, aiming to address key challenges in identifying subtle smoke patterns against varied backgrounds. Early detection is essential for prompt intervention to safeguard lives and property during potential fire incidents.

Key Objectives:

1. Algorithmic Diversity: Evaluate and compare the effectiveness of machine learning models (SVM, XGB, Logistic Regression, KNN, NN, Decision Trees, and Random Forests) in smoke detection.

2. Feature Selection Impact: Investigate how different feature selection techniques (variance thresholding, correlation analysis, feed-forward selection, k-best selection, Mutual Information Classification, Mean Absolute Difference) influence smoke detection model performance, aiming to optimize computational efficiency.

3. Benchmark Evaluation: Rigorously test and evaluate models on benchmark datasets, using metrics such as accuracy, precision, recall, and F1 score. Provide a comprehensive comparison to guide the selection of optimal models for specific smoke detection applications.

4. Practical Implications: Offer insights and recommendations for practitioners and researchers working on similar tasks, contributing not only to smoke detection technology but also to broader understanding of machine learning applications in critical domains.

Used Algorithms:

We provide a comprehensive overview of the machine learning models employed in the Smoke Detection task.

1. Random Forest:

Description: Random Forest is an ensemble learning method that constructs a multitude of decision trees during training and outputs the class that is the mode of the classes (classification) of the individual trees.

2. Logistic Regression:

Description: Logistic Regression is a linear model for binary classification that predicts the probability of an instance belonging to a particular class.

3. SVM (Support Vector Machine):

Description: Support Vector Machines are powerful classifiers that find the hyperplane that best separates instances of different classes in a high-dimensional space.

4. Naive Bayes:

Description: Naive Bayes is a probabilistic classifier based on Bayes' theorem with the "naive" assumption of independence between features.

5. K-Nearest Neighbors (KNN):

Description: K-Nearest Neighbors is a non-parametric and instance-based learning algorithm that classifies instances based on the majority class of their k-nearest neighbors.

6. Decision Tree:

Description: A decision tree is a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility.

7. AdaBoost:

Description: AdaBoost is an ensemble learning method that combines the predictions of weak classifiers to create a strong classifier.

8. Neural Network (Sigmoid):

Description: A Neural Network with a sigmoid activation function, representing a basic form of artificial neural networks for binary classification.

9. XGBoost:

Description: XGBoost is an optimized gradient-boosting algorithm designed for speed and performance, incorporating tree-based models.

Feature Selection Ways:

1. Variance Thresholding: Eliminates features with low variance, considering them less informative for the model.

2. Correlation Analysis: Identifies and removes highly correlated features to address multicollinearity and improve model interpretability.

3. Feed-Forward Selection: Iteratively adds features to the model based on their contribution, aiming to enhance predictive performance.

4. K-best Selection: Selects the k-best features based on statistical tests or scores, discarding less relevant variables for improved model simplicity.

5. Mutual Information Classification: Measures the mutual dependence between features and the target variable, aiding in feature selection for classification tasks.

6. Mean Absolute Difference: Evaluates the average absolute difference between feature values in different classes, helping identify discriminatory features.

7. Model Fit Selection for XGBoost, Random Forest, and Logistic Regression: Utilizes model-specific metrics to choose the most relevant features, optimizing performance for each algorithm.

Used Metrics:

1. Accuracy:

Definition: Accuracy is a measure of the overall correctness of the model. It calculates the ratio of correctly predicted instances to the total instances.

Equation:

$$Accuracy = \frac{True\ Positives\ +\ True\ Negatives}{Total\ Predictions}$$

Usefulness: While accuracy provides a general sense of how well the model performs, it might not be the best metric for imbalanced datasets.

2. Precision:

Definition: Precision measures the accuracy of positive predictions. It calculates the ratio of true positives to the total predicted positives.

Equation:

$$Precision = \frac{True\ Positives}{True\ Positives\ +\ False\ Positives}$$

Usefulness: Precision is crucial in scenarios where false positives are costly. In smoke classification, high precision means that when the model predicts a sample as smoke, it is likely to be correct.

3. Recall:

Definition: Recall measures the ability of the model to capture all the relevant instances. It calculates the ratio of true positives to the total actual positives.

Equation:

$$Recall = \frac{TruePositives}{TruePositives\ +\ FalseNegatives}$$

Recall is vital when the cost of false negatives is high. In the context of smoke classification, high recall indicates that the model is effective in identifying smoke cases, minimizing the chances of missing potentially dangerous things.

4. F1 Score:

Definition: The F1 score is the harmonic mean of precision and recall. It provides a balance between precision and recall.

Equation:

$$F1Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Usefulness: The F1 score is especially useful when there is an uneven class distribution. It considers both false positives and false negatives, making it a good overall metric for binary classification tasks like smoke cancer classification. A high F1 score indicates a model that performs well in terms of both precision and recall.

These metrics collectively offer insights into the model's performance. While accuracy gives a broad understanding, precision, recall, and f1 scores provide more nuanced insights into the model's ability to correctly identify snoke cases and avoid misclassifying benign cases.

Code:

```
model = LogisticRegression()

sfs = SFS(model,
          k_features=5,
          forward=True,
          floating=False,
          scoring='accuracy',
          cv=5)

sfs.fit(X_train, y_train)
print('Selected Features:', sfs.k_feature_names_)

accuracy = sfs.k_score_
print('Accuracy:', accuracy)

sfs_df = pd.DataFrame.from_dict(sfs.get_metric_dict()).T
sfs_df['avg_score'] = sfs_df['avg_score'].astype(float)
fig, ax = plt.subplots()
sfs_df.plot(kind='line', y='avg_score', ax=ax)
ax.set_xlabel('Number of Features')
ax.set_ylabel('Accuracy')
ax.set_title('Forward Selection Performance')
plt.show()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
X_train = X_train[['Humidity', 'TVOCppb', 'eCO2ppm', 'Raw Ethanol', 'CNT']]
```

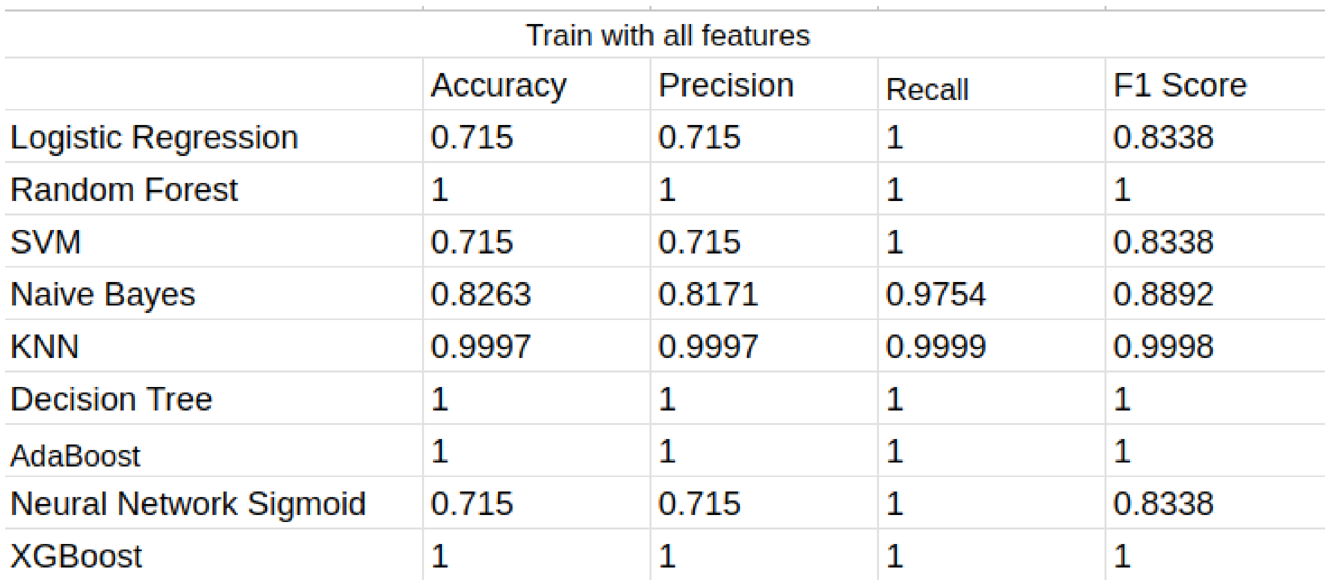
```
X_test = X_test[['Humidity', 'TVOCppb', 'eCO2ppm', 'Raw Ethanol', 'CNT']]


logreg_classifier = LogisticRegression()
svm_classifier = SVC()
nb_classifier = GaussianNB()
knn_classifier = KNeighborsClassifier()
dt_classifier = DecisionTreeClassifier()
ada_classifier = AdaBoostClassifier()
sigmoid_nn_classifier = MLPClassifier(activation='logistic')
softmax_nn_classifier = MLPClassifier(activation='softmax')
xgb_classifier = xgb.XGBClassifier()
rf_classifier = RandomForestClassifier()
classifiers = {
    'Random Forest': rf_classifier,
    'Logistic Regression': logreg_classifier,
    'SVM': svm_classifier,
    'Naive Bayes': nb_classifier,
    'K-Nearest Neighbors': knn_classifier,
    'Decision Tree': dt_classifier,
    'AdaBoost': ada_classifier,
    'Neural Network (Sigmoid)': sigmoid_nn_classifier,
    'XGBoost': xgb_classifier
}


accuracy_scores_train = []
precision_scores_train = []
```

```
recall_scores_train = []
f1_scores_train = []

accuracy_scores_test = []
precision_scores_test = []
recall_scores_test = []
f1_scores_test = []

for name, classifier in classifiers.items():
    print(f"Training and evaluating {name}...")
    classifier.fit(X_train, y_train)

    y_train_pred = classifier.predict(X_train)
    train_accuracy = accuracy_score(y_train, y_train_pred)
    train_precision = precision_score(y_train, y_train_pred)
    train_recall = recall_score(y_train, y_train_pred)
    train_f1 = f1_score(y_train, y_train_pred)

    y_test_pred = classifier.predict(X_test)
    test_accuracy = accuracy_score(y_test, y_test_pred)
    test_precision = precision_score(y_test, y_test_pred)
    test_recall = recall_score(y_test, y_test_pred)
    test_f1 = f1_score(y_test, y_test_pred)

    conf_matrix_train = confusion_matrix(y_train, y_train_pred)
    conf_matrix_test = confusion_matrix(y_test, y_test_pred)
```

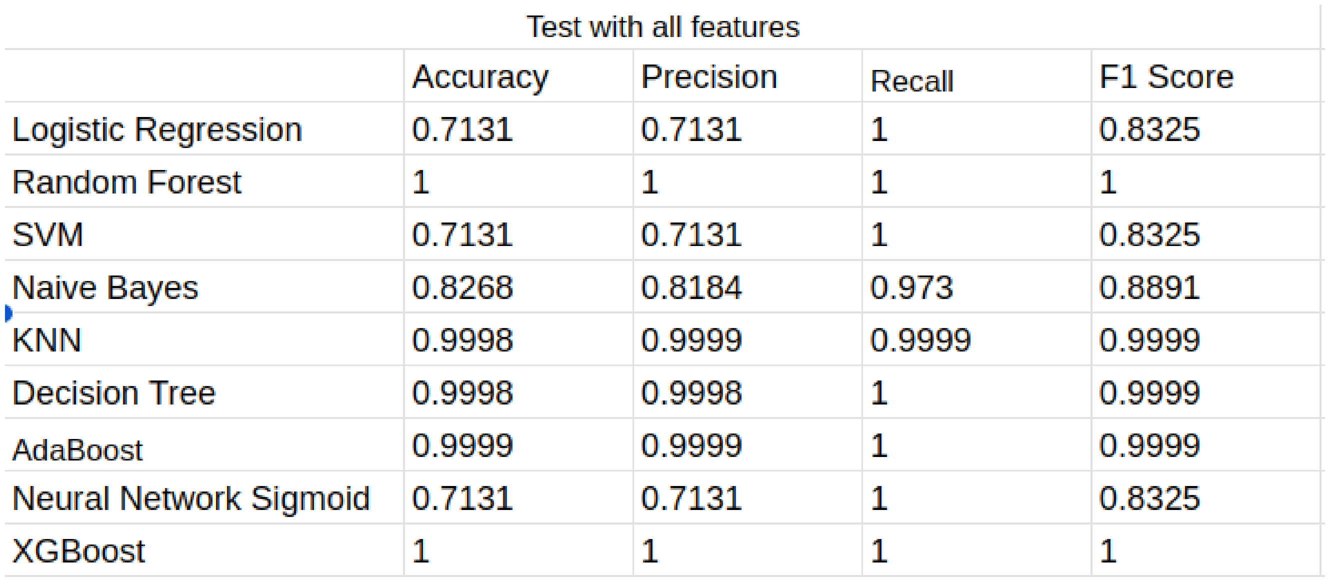
```
accuracy_scores_train.append(train_accuracy)
precision_scores_train.append(train_precision)
recall_scores_train.append(train_recall)
f1_scores_train.append(train_f1)

accuracy_scores_test.append(test_accuracy)
precision_scores_test.append(test_precision)
recall_scores_test.append(test_recall)
f1_scores_test.append(test_f1)

print("\nTraining Scores:")
print(f"Accuracy: {train_accuracy:.4f}")
print(f"Precision: {train_precision:.4f}")
print(f"Recall: {train_recall:.4f}")
print(f"F1 Score: {train_f1:.4f}")

print("\nTest Scores:")
print(f"Accuracy: {test_accuracy:.4f}")
print(f"Precision: {test_precision:.4f}")
print(f"Recall: {test_recall:.4f}")
print(f"F1 Score: {test_f1:.4f}")

plt.figure(figsize=(6, 5))
sns.heatmap(conf_matrix_train, annot=True, fmt="d", cmap="Blues", cbar=False,
            xticklabels=['Predicted 0', 'Predicted 1'],
                yticklabels=['Actual 0', 'Actual 1'])
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.title(f"{name} Training Confusion Matrix")
    plt.show()

    plt.figure(figsize=(6, 5))
    sns.heatmap(conf_matrix_test, annot=True, fmt="d", cmap="Blues", cbar=False,
                xticklabels=['Predicted 0', 'Predicted 1'],
                yticklabels=['Actual 0', 'Actual 1'])
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.title(f"{name} Test Confusion Matrix")
    plt.show()

score_names = ['Accuracy', 'Precision', 'Recall', 'F1 Score']
colors = ['blue', 'green', 'red', 'purple', 'orange', 'cyan', 'magenta', 'yellow', 'brown']

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(14, 10))
fig.suptitle('Training Scores', fontsize=16)

for i, score_list in enumerate([accuracy_scores_train, precision_scores_train,
recall_scores_train, f1_scores_train]):
    ax = axes[i // 2, i % 2]
    for j, (clf, score) in enumerate(zip(classifiers.keys(), score_list)):
        ax.bar(clf, score, color=colors[j % len(colors)], alpha=0.7, label=f'{clf}')
    ax.set_title(f'{score_names[i]} Scores')
    ax.set_ylim([0, 1])
    ax.set_xticklabels(classifiers.keys(), rotation=45, ha='right')
    ax.legend()

plt.tight_layout(rect=[0, 0, 1, 0.96])


fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(14, 10))
fig.suptitle('Test Scores', fontsize=16)


for i, score_list in enumerate([accuracy_scores_test, precision_scores_test,
recall_scores_test, f1_scores_test]):
    ax = axes[i // 2, i % 2]
    for j, (clf, score) in enumerate(zip(classifiers.keys(), score_list)):
        ax.bar(clf, score, color=colors[j % len(colors)], alpha=0.7, label=f'{clf}')

    ax.set_title(f'{score_names[i]} Scores')
    ax.set_ylim([0, 1])
    ax.set_xticklabels(classifiers.keys(), rotation=45, ha='right')
    ax.legend()

plt.tight_layout(rect=[0, 0, 1, 0.96])


plt.show()
```

<u>Output Screenshots:</u>

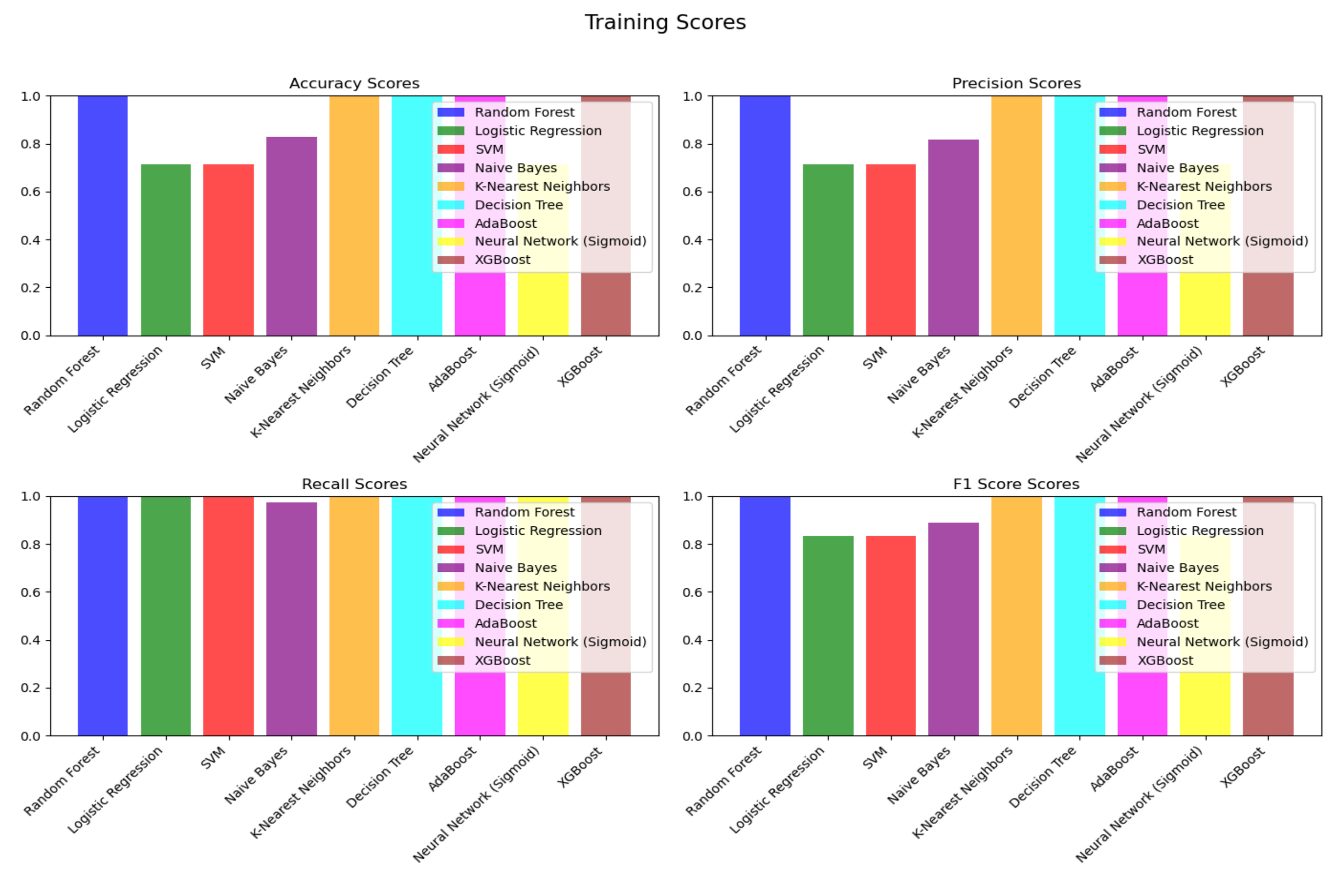
Before Feature Selection

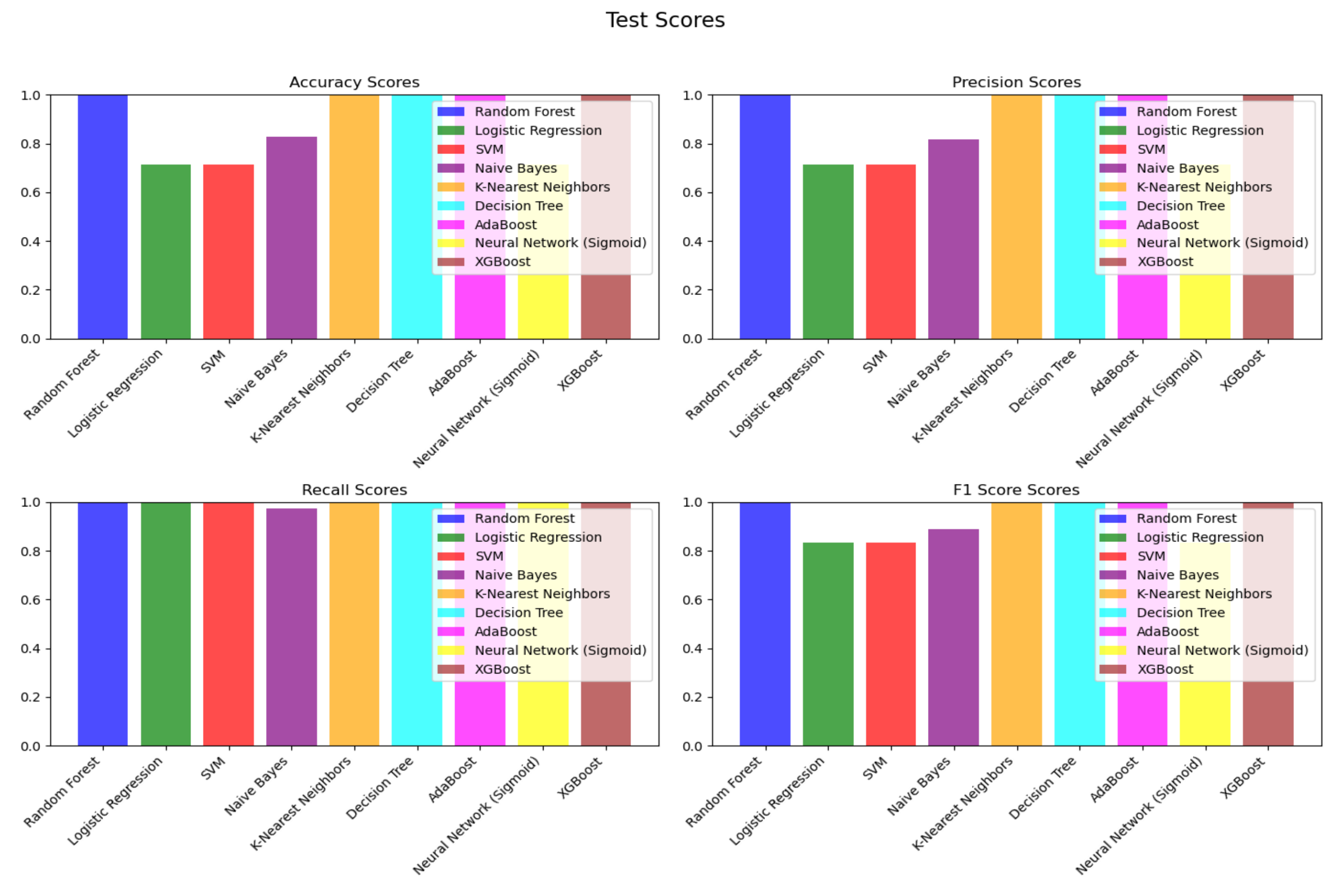


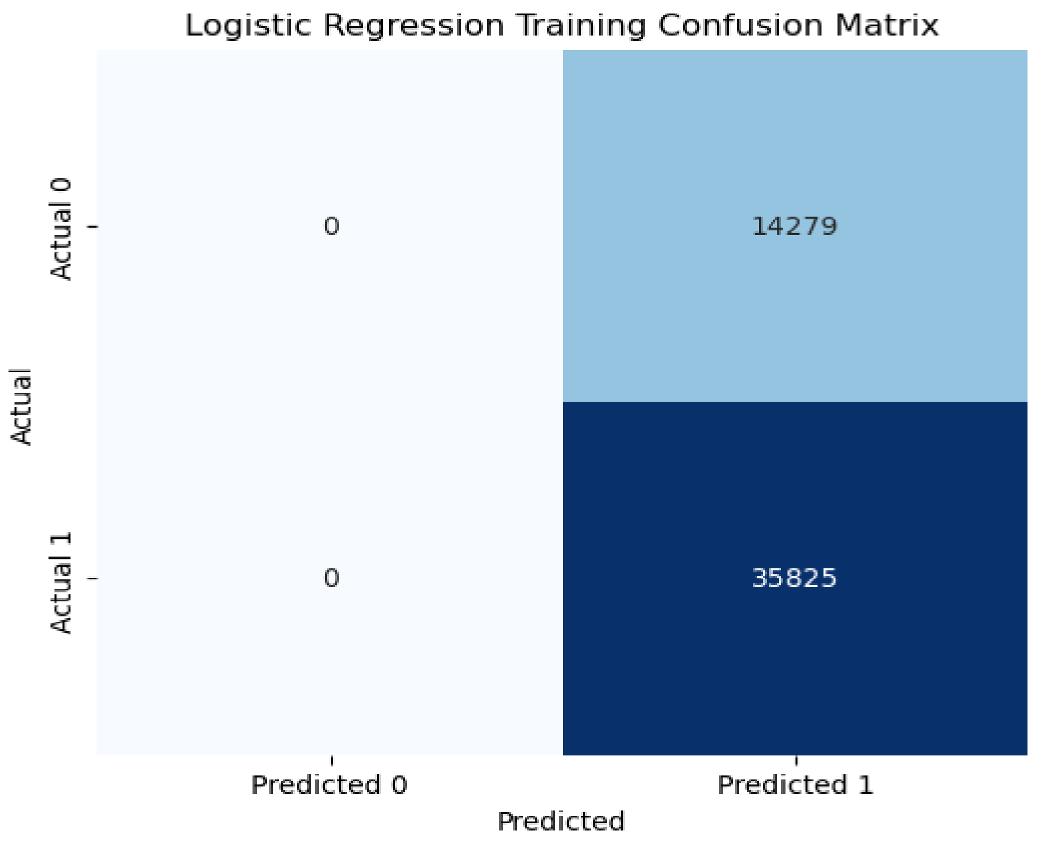
Train with all features				
	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.715	0.715	1	0.8338
Random Forest	1	1	1	1
SVM	0.715	0.715	1	0.8338
Naive Bayes	0.8263	0.8171	0.9754	0.8892
KNN	0.9997	0.9997	0.9999	0.9998
Decision Tree	1	1	1	1
AdaBoost	1	1	1	1
Neural Network Sigmoid	0.715	0.715	1	0.8338
XGBoost	1	1	1	1

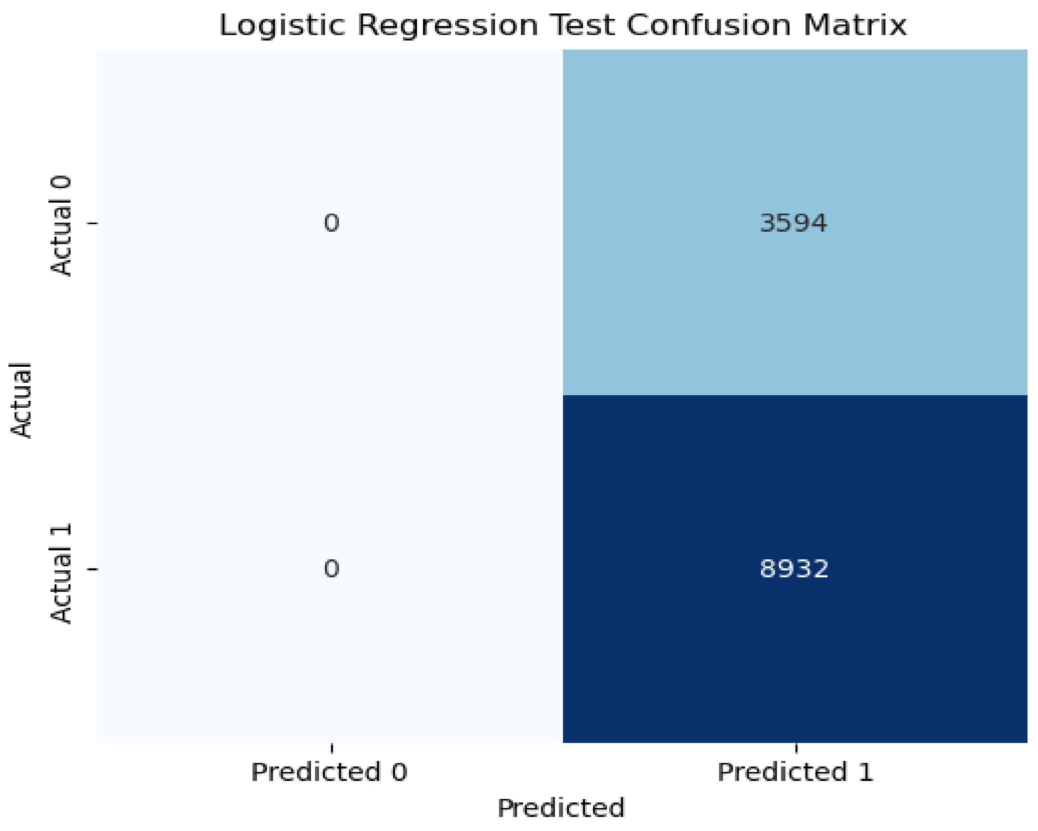


Test with all features				
	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.7131	0.7131	1	0.8325
Random Forest	1	1	1	1
SVM	0.7131	0.7131	1	0.8325
Naive Bayes	0.8268	0.8184	0.973	0.8891
KNN	0.9998	0.9999	0.9999	0.9999
Decision Tree	0.9998	0.9998	1	0.9999
AdaBoost	0.9999	0.9999	1	0.9999
Neural Network Sigmoid	0.7131	0.7131	1	0.8325
XGBoost	1	1	1	1

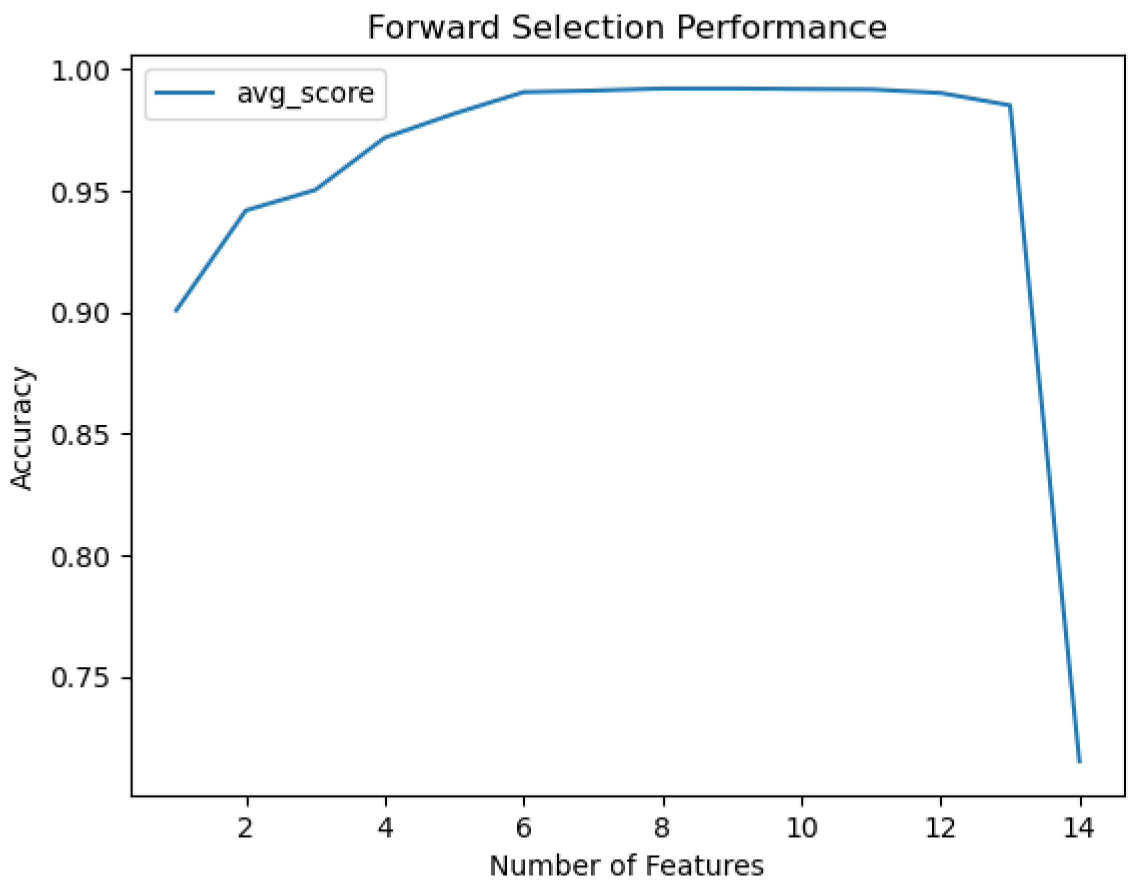

Training Scores
Accuracy Scores
Precision Scores
Recall Scores
F1 Score Scores
Random Forest
Logistic Regression
SVM
Naive Bayes
K-Nearest Neighbors
Decision Tree
AdaBoost
Neural Network (Sigmoid)
XGBoost
0.0
0.2
0.4
0.6
0.8
1.0


Test Scores
Accuracy Scores
Precision Scores
Recall Scores
F1 Score Scores
Random Forest
Logistic Regression
SVM
Naive Bayes
K-Nearest Neighbors
Decision Tree
AdaBoost
Neural Network (Sigmoid)
XGBoost
0.0
0.2
0.4
0.6
0.8
1.0

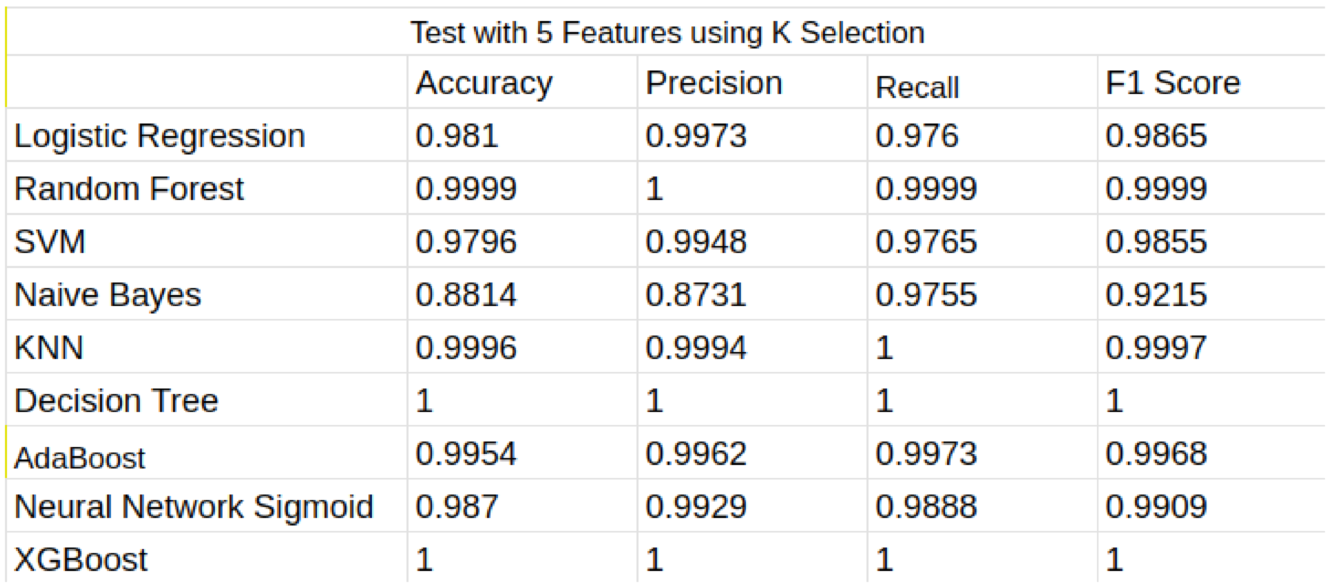

Logistic Regression Training Confusion Matrix
Actual 0
Actual 1
Actual
0
14279
0
35825
Predicted 0
Predicted 1
Predicted


Logistic Regression Test Confusion Matrix
Actual 0
Actual 1
Actual
0
3594
0
8932
Predicted 0
Predicted 1
Predicted

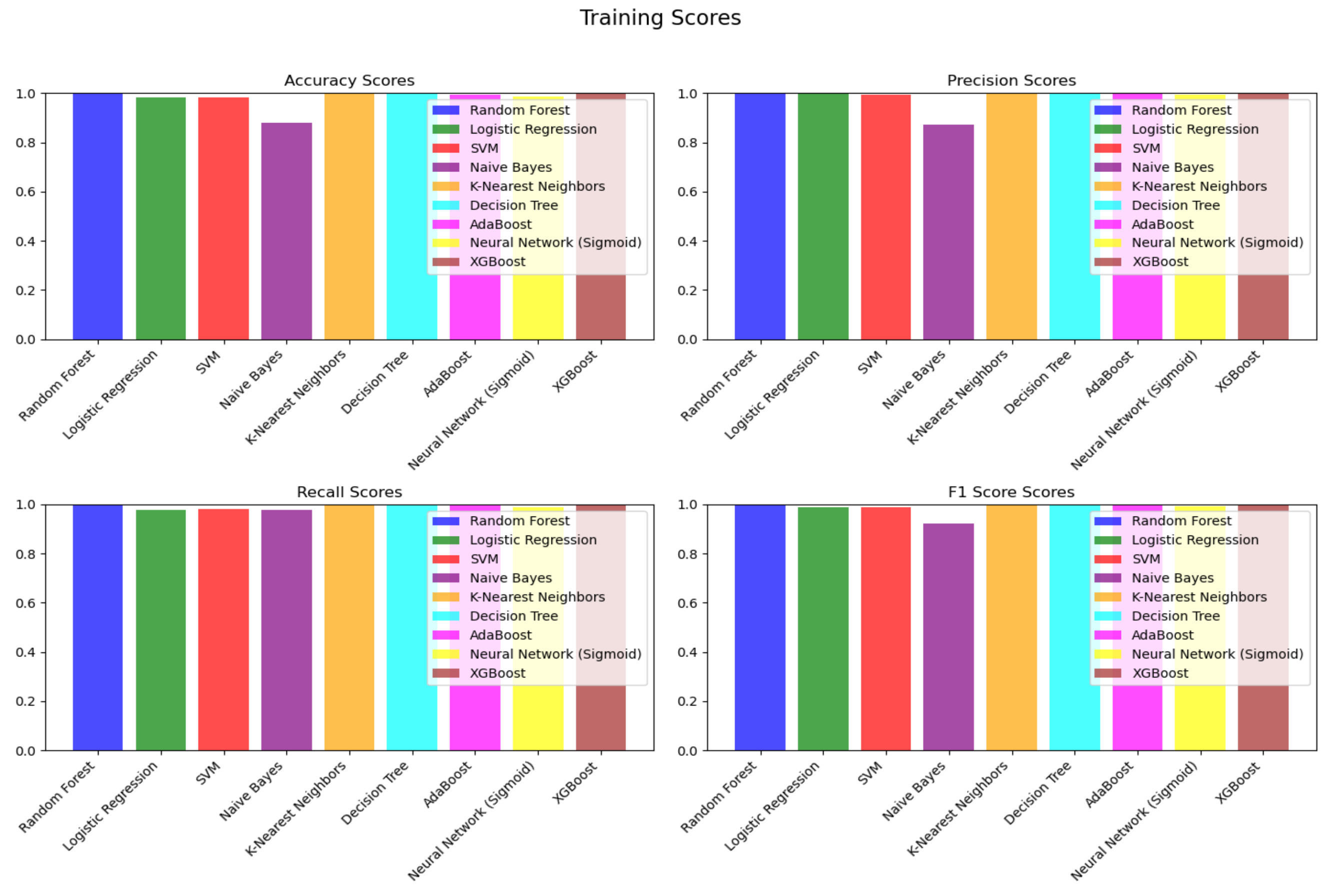
After Feature Selection

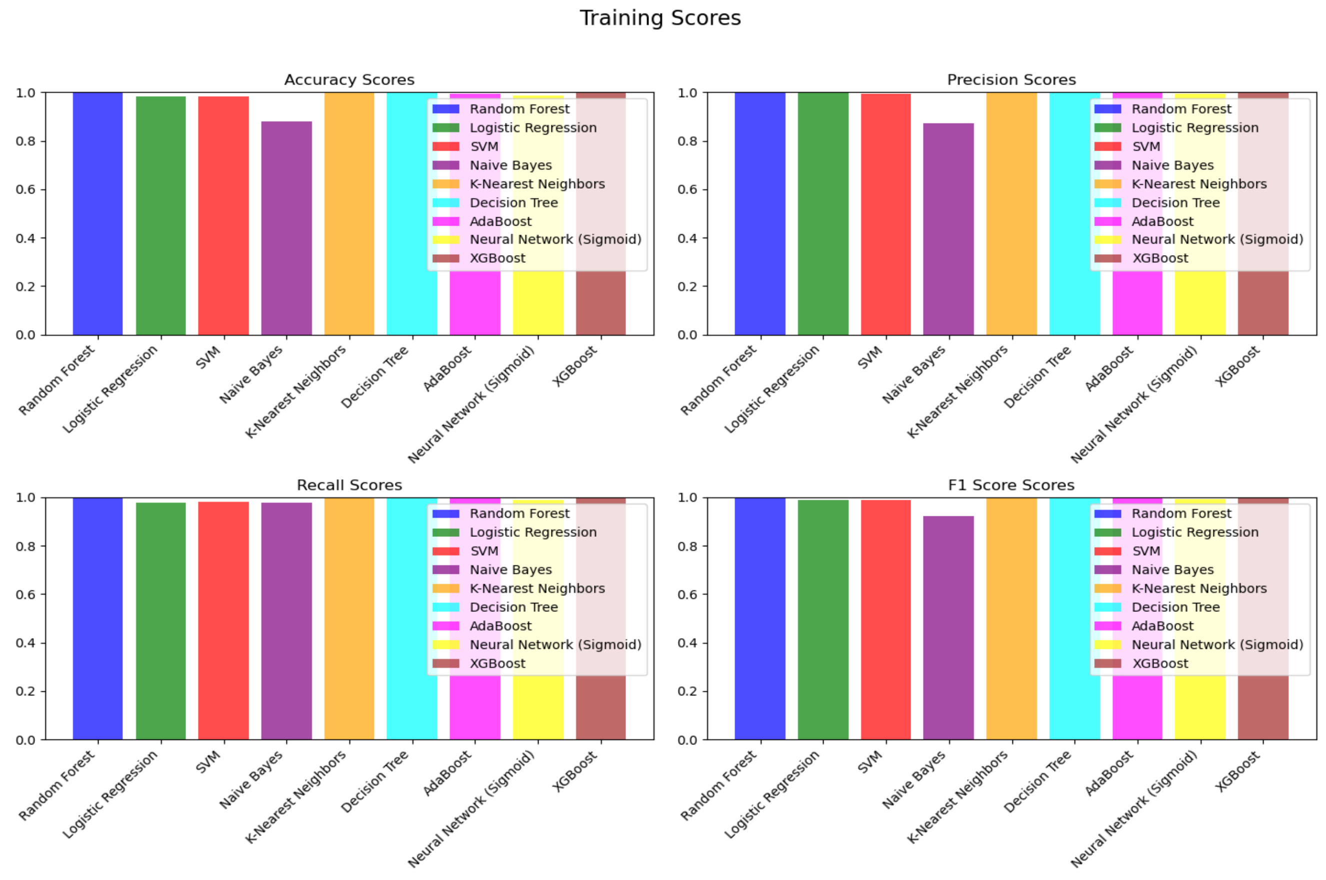


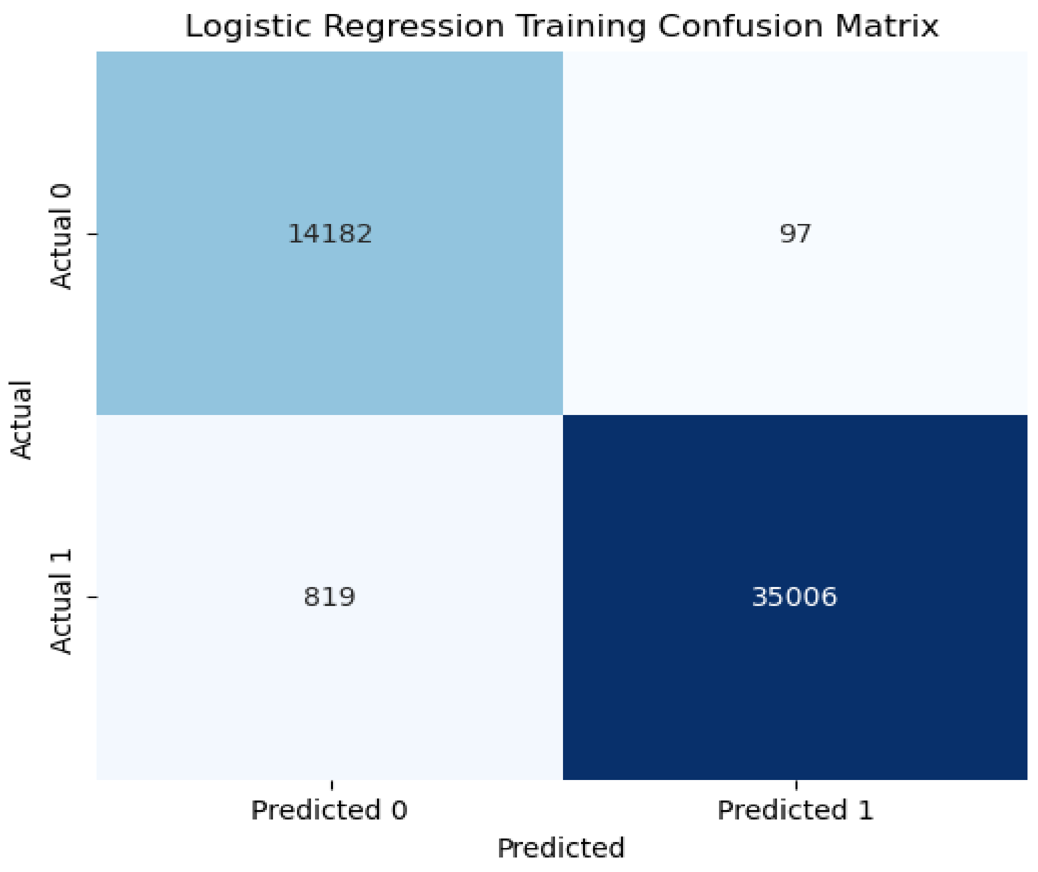
Train with 5 Features using K Selection				
	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.9817	0.9972	0.9771	0.9871
Random Forest	1	1	1	1
SVM	0.9813	0.9946	0.9791	0.9868
Naive Bayes	0.8799	0.8708	0.977	0.9208
KNN	0.9997	0.9996	1	0.9998
Decision Tree	0.9999	0.9999	1	0.9999
AdaBoost	0.9956	0.9967	0.9972	0.9969
Neural Network Sigmoid	0.9865	0.9925	0.9886	0.9906
XGBoost	1	1	1	1

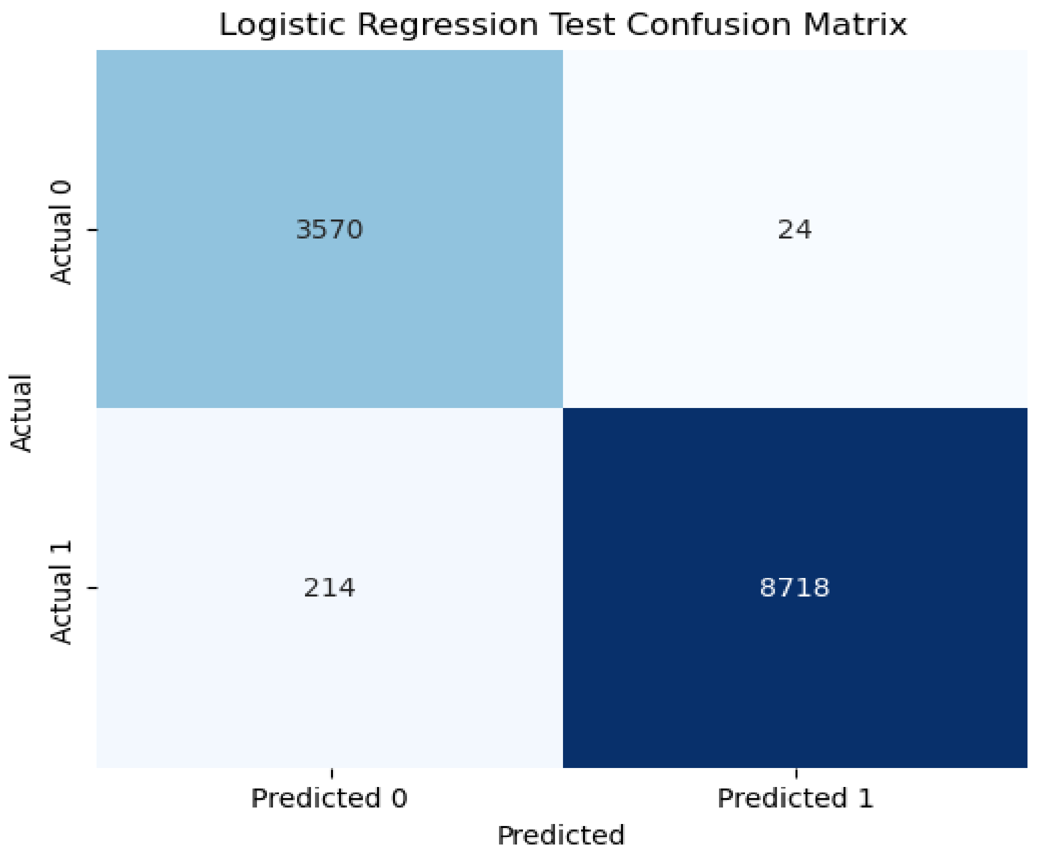


Test with 5 Features using K Selection				
	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.981	0.9973	0.976	0.9865
Random Forest	0.9999	1	0.9999	0.9999
SVM	0.9796	0.9948	0.9765	0.9855
Naive Bayes	0.8814	0.8731	0.9755	0.9215
KNN	0.9996	0.9994	1	0.9997
Decision Tree	1	1	1	1
AdaBoost	0.9954	0.9962	0.9973	0.9968
Neural Network Sigmoid	0.987	0.9929	0.9888	0.9909
XGBoost	1	1	1	1


Training Scores
Accuracy Scores
Precision Scores
Recall Scores
F1 Score Scores
Random Forest
Logistic Regression
SVM
Naive Bayes
K-Nearest Neighbors
Decision Tree
AdaBoost
Neural Network (Sigmoid)
XGBoost
0.0
0.2
0.4
0.6
0.8
1.0


Training Scores
Accuracy Scores
Precision Scores
Recall Scores
F1 Score Scores
Random Forest
Logistic Regression
SVM
Naive Bayes
K-Nearest Neighbors
Decision Tree
AdaBoost
Neural Network (Sigmoid)
XGBoost
0.0
0.2
0.4
0.6
0.8
1.0


Logistic Regression Training Confusion Matrix
Actual
Actual 0
Actual 1
14182
97
819
35006
Predicted 0
Predicted 1
Predicted


Logistic Regression Test Confusion Matrix
Actual
Actual 0
Actual 1
3570
24
214
8718
Predicted 0
Predicted 1
Predicted