**WEEK 2 REPORT**

**TOPIC : STUDY OF THE REFERENCE-CODE**

**Data Exploration**:

* The data is loaded from an Excel sheet into a pandas DataFrame.
* Initial exploration is performed by checking the first few rows, shape, and data types of the DataFrame.
* The dataset contains both numerical and categorical features, with the Section column identified as a categorical feature containing string values ('R', 'B', 'F').
* The categorical feature Section is converted to numerical values using the .cat.codes method.
* Missing values are checked, and the data is confirmed to have no missing entries.

**Correlation Analysis**:

* A correlation matrix is computed for the numerical features, and a heatmap is plotted to visualize the correlation.
* Correlation heatmaps are created for correlations above or below 0.5 to identify significant relationships between features.

**Data Visualization**:

* Pairplots are generated to visualize relationships between the selected quantitative features and the target variable FailureMode.
* A stacked bar plot is created to explore the relationship between the Section type and FailureMode.
* A count plot is used to check the distribution of the FailureMode, revealing an imbalanced dataset.

**Data Preprocessing**:

* One-hot encoding is applied to the Section column to convert categorical values into binary variables.
* The dataset is split into features (X) and the target variable (y).

**Implementation of a Naive Bayes classifier :**

**using the GaussianNB class from the sklearn.naive\_bayes module in Python.** The classifier is trained and evaluated on a dataset, with results including accuracy, confusion matrices, and classification reports.

### **Key Outputs:**

1. **Accuracy on Training and Test Sets:**
   * **Training Set Accuracy:** 69%
   * **Test Set Accuracy:** 76%

**Confusion Matrix for Training Set:** The confusion matrix shows the number of correct and incorrect predictions for each class:  
  
  
[[82 17 5 2]

[15 39 12 0]

[ 2 20 61 1]

[ 4 3 5 7]]

1. **Classification Report for Training Set:**
   * **Class 1:** Precision = 0.80, Recall = 0.77, F1-Score = 0.78
   * **Class 2:** Precision = 0.49, Recall = 0.59, F1-Score = 0.54
   * **Class 3:** Precision = 0.73, Recall = 0.73, F1-Score = 0.73
   * **Class 4:** Precision = 0.70, Recall = 0.37, F1-Score = 0.48
   * **Overall Accuracy:** 69%

**Confusion Matrix for Test Set:** The confusion matrix for the test set reflects the classifier's performance on unseen data:  
  
[[38 4 3 1]

[ 6 21 3 0]

[ 2 5 30 1]

[ 1 1 1 1]]

1. **Classification Report for Test Set:**
   * **Class 1:** Precision = 0.81, Recall = 0.83, F1-Score = 0.82
   * **Class 2:** Precision = 0.68, Recall = 0.70, F1-Score = 0.69
   * **Class 3:** Precision = 0.81, Recall = 0.79, F1-Score = 0.80
   * **Class 4:** Precision = 0.33, Recall = 0.25, F1-Score = 0.29
   * **Overall Accuracy:** 76%

### **Interpretation:**

* The classifier performs reasonably well, with higher accuracy on the test set than the training set.
* **Class 1** and **Class 3** have higher precision and recall, indicating that the classifier is better at predicting these classes.
* **Class 4** has the lowest performance, likely due to a smaller support count (only 4 instances in the test set).

This output provides a comprehensive evaluation of the Naive Bayes model, showing both its strengths and areas for improvement.

**Implementation of a K-Nearest Neighbors (K-NN) classifier**

using the KNeighborsClassifier class from the sklearn.neighbors module. The classifier is trained and evaluated on a dataset, with results including accuracy, confusion matrices, and classification reports.

### **Key Outputs:**

1. **Accuracy on Training and Test Sets:**
   * **Training Set Accuracy:** 80%
   * **Test Set Accuracy:** 85%

**Confusion Matrix for Training Set:** The confusion matrix shows the number of correct and incorrect predictions for each class:  
  
[[96 5 3 2]

[15 46 5 0]

[ 4 8 72 0]

[ 4 3 6 6]]

1. **Classification Report for Training Set:**
   * **Class 1:** Precision = 0.81, Recall = 0.91, F1-Score = 0.85
   * **Class 2:** Precision = 0.74, Recall = 0.70, F1-Score = 0.72
   * **Class 3:** Precision = 0.84, Recall = 0.86, F1-Score = 0.85
   * **Class 4:** Precision = 0.75, Recall = 0.32, F1-Score = 0.44
   * **Overall Accuracy:** 80%

**Confusion Matrix for Test Set:** The confusion matrix for the test set reflects the classifier's performance on unseen data:  
  
[[43 1 0 2]

[ 7 20 3 0]

[ 1 3 34 0]

[ 0 0 1 3]]

1. **Classification Report for Test Set:**
   * **Class 1:** Precision = 0.84, Recall = 0.93, F1-Score = 0.89
   * **Class 2:** Precision = 0.83, Recall = 0.67, F1-Score = 0.74
   * **Class 3:** Precision = 0.89, Recall = 0.89, F1-Score = 0.89
   * **Class 4:** Precision = 0.60, Recall = 0.75, F1-Score = 0.67
   * **Overall Accuracy:** 85%

### **Interpretation:**

* The K-NN classifier performs well, with higher accuracy on both the training and test sets compared to the Naive Bayes classifier.
* **Class 1** and **Class 3** show high precision, recall, and F1-scores, indicating the model's effectiveness in predicting these classes.
* **Class 4** has lower performance in the training set but shows improvement in the test set, likely due to a small number of instances (only 4 in the test set).
* The model generalizes well, as indicated by the similar accuracy scores on the training and test sets.

This output shows that the K-NN model provides strong predictive performance, particularly for the more frequent classes.

**Implementation of a Decision Tree classifier**

using the DecisionTreeClassifier class from the sklearn.tree module. The classifier is trained and evaluated on a dataset, with results including accuracy, confusion matrices, and classification reports.

### **Key Outputs:**

1. **Accuracy on Training and Test Sets:**
   * **Training Set Accuracy:** 100%
   * **Test Set Accuracy:** 79%

**Confusion Matrix for Training Set:** The confusion matrix shows perfect classification on the training set:  
  
[[106 0 0 0]

[ 0 66 0 0]

[ 0 0 84 0]

[ 0 0 0 19]]

1. **Classification Report for Training Set:**
   * **All Classes:** Precision = 1.00, Recall = 1.00, F1-Score = 1.00
   * **Overall Accuracy:** 100%

**Confusion Matrix for Test Set:** The confusion matrix for the test set reflects the classifier's performance on unseen data:  
  
[[37 6 3 0]

[ 4 22 1 3]

[ 4 1 31 2]

[ 0 1 0 3]]

1. **Classification Report for Test Set:**
   * **Class 1:** Precision = 0.82, Recall = 0.80, F1-Score = 0.81
   * **Class 2:** Precision = 0.73, Recall = 0.73, F1-Score = 0.73
   * **Class 3:** Precision = 0.89, Recall = 0.82, F1-Score = 0.85
   * **Class 4:** Precision = 0.38, Recall = 0.75, F1-Score = 0.50
   * **Overall Accuracy:** 79%

### **Interpretation:**

* **Training Set:** The Decision Tree classifier achieves 100% accuracy on the training set, indicating that the model has perfectly memorized the training data. This is reflected in the confusion matrix, where all instances are correctly classified.
* **Test Set:** The accuracy drops to 79% on the test set, which is a common sign of overfitting, where the model performs well on the training data but struggles to generalize to unseen data.
* **Class-wise Performance:**
  + **Class 1** and **Class 3** show strong performance with high precision, recall, and F1-scores on the test set.
  + **Class 2** has consistent performance across both precision and recall.
  + **Class 4** shows a lower precision but higher recall, suggesting that the model is more cautious in predicting this class.

### **Conclusion:**

The Decision Tree classifier provides excellent performance on the training set, but the drop in test accuracy indicates overfitting. While the model performs well on the more frequent classes (Classes 1, 2, and 3), it struggles with the less frequent class (Class 4). To improve generalization, techniques such as pruning, adjusting the maximum depth, or using ensemble methods (like Random Forests) could be considered.

**Implementation of a Random Forest classifier**

using the RandomForestClassifier from the sklearn.ensemble module, followed by hyperparameter tuning to optimize its performance. Below is a breakdown of the key outputs and results:

### **Key Outputs:**

1. **Initial Random Forest Classifier:**
   * **Training Set Accuracy:** 100%
   * **Test Set Accuracy:** 86%

**Confusion Matrix for Training Set (Initial Model):** The confusion matrix shows perfect classification on the training set:  
  
[[106 0 0 0]

[ 0 66 0 0]

[ 0 0 84 0]

[ 0 0 0 19]]

**Confusion Matrix for Test Set (Initial Model):**  
[[43 2 0 1]

[ 5 21 3 1]

[ 2 2 34 0]

[ 0 0 0 4]]

1. **Classification Report for Test Set (Initial Model):**
   * **Class 1:** Precision = 0.86, Recall = 0.93, F1-Score = 0.90
   * **Class 2:** Precision = 0.84, Recall = 0.70, F1-Score = 0.76
   * **Class 3:** Precision = 0.92, Recall = 0.89, F1-Score = 0.91
   * **Class 4:** Precision = 0.67, Recall = 1.00, F1-Score = 0.80
   * **Overall Accuracy:** 86%
2. **Feature Importances (Initial Model):**
   * A horizontal bar plot shows the relative importance of each feature, helping identify which features are most influential in the model.

### **Hyperparameter Tuning with Randomized Search:**

* The code then performs hyperparameter tuning using RandomizedSearchCV to explore a range of parameters, such as the number of trees, maximum depth, and the method of selecting samples.

**Best Parameters Found:**  
{'n\_estimators': 1000, 'min\_samples\_split': 2, 'min\_samples\_leaf': 1, 'max\_features': 'auto', 'max\_depth': 50, 'bootstrap': False}

1. **Random Forest Classifier After Tuning:**
   * **Training Set Accuracy:** 100%
   * **Test Set Accuracy:** 86%
   * The accuracy remains unchanged after tuning, indicating that the initial model was already well-optimized.

**Confusion Matrix for Training Set (After Tuning):**  
[[106 0 0 0]

[ 0 66 0 0]

[ 0 0 84 0]

[ 0 0 0 19]]

**Confusion Matrix for Test Set (After Tuning):**  
[[42 3 0 1]

[ 5 21 3 1]

[ 2 1 35 0]

[ 0 0 0 4]]

1. **Classification Report for Test Set (After Tuning):**
   * **Class 1:** Precision = 0.86, Recall = 0.91, F1-Score = 0.88
   * **Class 2:** Precision = 0.84, Recall = 0.70, F1-Score = 0.76
   * **Class 3:** Precision = 0.92, Recall = 0.92, F1-Score = 0.92
   * **Class 4:** Precision = 0.67, Recall = 1.00, F1-Score = 0.80
   * **Overall Accuracy:** 86%
2. **Feature Importances (After Tuning):**
   * Another bar plot is generated to show the feature importances for the tuned model.

### **Conclusion:**

* The Random Forest classifier shows strong performance on both the training and test sets, with 100% accuracy on the training set and 86% accuracy on the test set.
* The hyperparameter tuning did not improve the test accuracy, suggesting that the initial model was already well-optimized.
* The feature importance plots provide insights into which features are most influential, which could guide further analysis or feature engineering efforts.

### **XGBoost Classifier Results:**

* **Training Accuracy:** 1.00 (indicates overfitting)
* **Test Accuracy:** 0.83 (strong performance, despite overfitting)

#### **Confusion Matrix and Classification Report:**

* **Training Set:**
  + **All classes** achieved 100% precision, recall, and F1-score, indicating perfect classification but also potential overfitting.
* **Test Set:**
  + **Class 1:** Precision = 0.82, Recall = 0.89, F1-Score = 0.85
  + **Class 2:** Precision = 0.76, Recall = 0.63, F1-Score = 0.69
  + **Class 3:** Precision = 0.92, Recall = 0.89, F1-Score = 0.91
  + **Class 4:** Precision = 0.67, Recall = 1.00, F1-Score = 0.80
  + **Overall Accuracy:** 0.83

### **LightGBM Classifier Results:**

* **Training Accuracy:** 1.00 (indicates overfitting)
* **Test Accuracy:** 0.80 (slightly lower than XGBoost but still strong)

#### **Confusion Matrix and Classification Report:**

* **Training Set:**
  + Similar to XGBoost, **all classes** achieved 100% precision, recall, and F1-score, indicating potential overfitting.
* **Test Set:**
  + **Class 1:** Precision = 0.80, Recall = 0.85, F1-Score = 0.82
  + **Class 2:** Precision = 0.66, Recall = 0.63, F1-Score = 0.64
  + **Class 3:** Precision = 0.91, Recall = 0.84, F1-Score = 0.88
  + **Class 4:** Precision = 0.80, Recall = 1.00, F1-Score = 0.89
  + **Overall Accuracy:** 0.80

### **Key Observations:**

1. **Overfitting:** Both XGBoost and LightGBM exhibit overfitting, as indicated by the perfect scores on the training set but a drop in accuracy on the test set. This is a common issue with powerful ensemble methods when not properly regularized.
2. **Test Performance:** XGBoost performed slightly better on the test set compared to LightGBM, with an accuracy of 0.83 vs. 0.80. Both models, however, significantly outperformed AdaBoost and Random Forest.
3. **Class-Specific Performance:**
   * **Class 3** consistently shows high precision, recall, and F1-scores across all models.
   * **Class 4** has high recall in both XGBoost and LightGBM, but precision and F1-scores vary slightly, indicating some instability in the prediction.

### **AdaBoost Classifier Results:**

* **Training Accuracy:** 0.76
* **Test Accuracy:** 0.67

### **Confusion Matrix and Classification Report:**

#### **Training Set:**

* **Class 1:** Precision = 0.81, Recall = 0.80, F1-Score = 0.81
* **Class 2:** Precision = 0.59, Recall = 0.70, F1-Score = 0.64
* **Class 3:** Precision = 0.84, Recall = 0.74, F1-Score = 0.78
* **Class 4:** Precision = 0.83, Recall = 0.79, F1-Score = 0.81
* **Overall Accuracy:** 0.76

#### **Test Set:**

* **Class 1:** Precision = 0.73, Recall = 0.78, F1-Score = 0.76
* **Class 2:** Precision = 0.48, Recall = 0.50, F1-Score = 0.49
* **Class 3:** Precision = 0.81, Recall = 0.68, F1-Score = 0.74
* **Class 4:** Precision = 0.33, Recall = 0.50, F1-Score = 0.40
* **Overall Accuracy:** 0.67

### **Key Observations:**

* The **AdaBoost classifier** has lower accuracy on both the training and test sets compared to the **Random Forest** model.
* **Class 3** seems to perform relatively well across both models, but there is a significant drop in performance for **Class 4** on the test set with the AdaBoost model.
* The **precision, recall, and F1-scores** indicate that the model may struggle more with correctly predicting certain classes, especially in an imbalanced setting.

### **CatBoost Classifier Results:**

* **Training Accuracy:** 1.00 (indicates overfitting, as seen in other models)
* **Test Accuracy:** 0.84 (strong performance, comparable to XGBoost)

#### **Confusion Matrix and Classification Report:**

* **Training Set:**
  + **All classes** achieved 100% precision, recall, and F1-score, suggesting perfect classification on the training data but potential overfitting.
* **Test Set:**
  + **Class 1:** Precision = 0.85, Recall = 0.87, F1-Score = 0.86
  + **Class 2:** Precision = 0.75, Recall = 0.70, F1-Score = 0.72
  + **Class 3:** Precision = 0.92, Recall = 0.89, F1-Score = 0.91
  + **Class 4:** Precision = 0.67, Recall = 1.00, F1-Score = 0.80
  + **Overall Accuracy:** 0.84

### **Key Observations:**

1. **Overfitting:** CatBoost, like the other models, shows overfitting on the training data. The model achieves perfect accuracy on the training set, which is typical for powerful models when hyperparameters are not tuned or regularization is not applied.
2. **Test Performance:** The performance on the test set is very strong, with an accuracy of 0.84. This places CatBoost slightly ahead of LightGBM but behind XGBoost, which had a test accuracy of 0.83. The differences in performance among these models are relatively small and might depend on the specific characteristics of the data.
3. **Class-Specific Performance:**
   * **Class 3** has high precision and recall across all models, indicating that the model is consistently good at predicting this class.
   * **Class 2** shows a slightly lower precision and recall compared to Class 3 but performs better than Class 4, indicating some difficulty in distinguishing this class.

## **From the models, Random Forest is the one having highest accuracy**

Tharun Varada, Eswari B

7386350253 , 89192 27494

30-08-2024