**Project on credit card fraud Prediction**

**CLASSIFCAION MODELS:**

1. **LOGISTIC REGRESSION**

Logistic Regression is a statistical method used for binary classification problems. Despite its name, it's a technique for classification rather than regression. It models the probability of a certain class or event existing (e.g., fraud vs. non-fraud in credit card transactions, spam vs. non-spam emails) based on one or more independent variables (features).

**Key Aspects of Logistic Regression:**

* **Model Function:**
* Logistic Regression models the probability that a given input belongs to a certain class using the logistic function (also known as the sigmoid function). The output of the logistic function is constrained between 0 and
* **Binary Classification:**
  + It's mainly used for binary classification tasks (classifying into two classes), though modifications like multinomial logistic regression handle multi-class classification.
* **Decision Boundary:**
  + Logistic Regression separates classes using a decision boundary. For a binary problem, the boundary is a linear function of the input features. Above the boundary, it predicts one class, and below it predicts the other class.
* **Cost Function (Optimization):**
  + Logistic Regression uses the concept of maximum likelihood estimation. It minimizes a cost function (often the cross-entropy loss) to find the best-fitting model parameters (weights and bias) that minimize the error between predicted and actual outcomes.
* **Probabilistic Interpretation:**
  + Logistic Regression outputs probabilities. After obtaining the linear combination of input features with weights (dot product) plus a bias term, it passes this through the logistic (sigmoid) function to get a probability value between 0 and 1.
* **Assumptions:**
  + Assumes a linear relationship between the independent variables and the log-odds of the dependent variable.
  + Assumes little to no multicollinearity among independent variables.
  + Assumes the absence of outliers in the data.

**Workflow of Logistic Regression:**

1. **Data Preprocessing:**
   * Handling missing values, encoding categorical variables, feature scaling, etc.
2. **Splitting Data:**
   * Splitting the dataset into training and testing sets.
3. **Model Training:**
   * Using the training set to fit the Logistic Regression model by optimizing its parameters.
4. **Model Evaluation:**
   * Assessing the model's performance using evaluation metrics like accuracy, precision, recall, F1-score, ROC curve, etc., on the test set.

**Advantages of Logistic Regression:**

* Computationally efficient.
* Provides probabilities for outcomes.
* Robust to noise and can handle overfitting with proper regularization.

**Limitations:**

* Assumes a linear relationship between features and log-odds.
* May not perform well with non-linear relationships.
* Sensitive to outliers.
* Doesn't handle a large number of categorical features or missing values easily.

Logistic Regression serves as a fundamental algorithm in the realm of classification and is widely used due to its simplicity, interpretability, and effectiveness in many practical scenarios.

1. **SUPPORT VECTOR MACHINE(SVM)**

Support Vector Machines (SVM) is a powerful supervised machine learning algorithm used for classification and regression tasks. It's highly effective in high-dimensional spaces and is versatile in handling both linear and non-linear data classification. Here's a detailed explanation of SVM:

**Basic Concept:**

SVM is primarily used for classification tasks and aims to find the best hyperplane that separates classes with the largest margin.

**Key Aspects of SVM:**

* **Maximum Margin:**
  + SVM finds the hyperplane that maximizes the margin, which is the distance between the hyperplane and the nearest data points (support vectors) from each class.
  + These support vectors are the data points closest to the decision boundary and significantly influence the position and orientation of the hyperplane.
* **Kernel Trick for Non-Linearity:**
  + SVM can handle non-linear data by mapping the input features into a higher-dimensional space using kernel functions (e.g., polynomial, radial basis function (RBF), sigmoid, etc.).
  + The kernel function calculates the dot product of transformed feature vectors in this higher-dimensional space without actually transforming the data, allowing for non-linear decision boundaries.
* **Regularization (C parameter):**
  + The C parameter in SVM trades off the classification of training points against the maximization of the margin. A smaller C value allows more misclassifications but a wider margin, while a larger C value may lead to a narrower margin but fewer misclassifications.
* **Margin and Loss Function:**
  + The hinge loss function is used to penalize misclassifications in SVM.
  + SVM aims to minimize this hinge loss while maximizing the margin.
* **Kernel Selection:**
  + Choosing the appropriate kernel function is crucial in SVM. It depends on the data characteristics and finding the best kernel involves some trial and error.

**Workflow of SVM:**

1. **Data Preprocessing:**
   * Similar to other machine learning algorithms, data preprocessing steps like normalization, standardization, handling missing values, and feature engineering might be required.
2. **Model Training:**
   * SVM learns the optimal hyperplane that best separates classes based on the training data. The optimization problem is solved using mathematical techniques like the Sequential Minimal Optimization (SMO) algorithm.
3. **Kernel Selection and Tuning:**
   * Experimentation with different kernel functions and tuning hyperparameters like C, gamma (for RBF), degree (for polynomial), etc., to improve model performance.
4. **Model Evaluation:**
   * Assessing the model's performance using evaluation metrics such as accuracy, precision, recall, F1-score, ROC curve, etc., on a test dataset.

**Advantages of SVM:**

* Effective in high-dimensional spaces.
* Versatile due to kernel trick for handling non-linearity.
* Robust against overfitting, especially in high-dimensional space.
* Works well with a clear margin of separation.

**Limitations:**

* Computationally expensive for large datasets.
* Choosing the appropriate kernel and tuning hyperparameters can be challenging.
* Less effective when the number of features is much greater than the number of samples.

SVM remains a widely used algorithm in many fields due to its ability to handle complex decision boundaries, especially in scenarios where the data has clear class separations or when non-linear relationships exist between features and classes.

1. **DESCISION TREE CLASSIFIER**

A Decision Tree Classifier is a supervised learning algorithm used for both classification and regression tasks. It predicts the class label or target value of an instance by learning simple decision rules inferred from the features. It organizes a set of data into a tree-like structure by partitioning it into smaller subsets, ultimately leading to a leaf node representing the class label or value.

**Basic Concept:**

* Decision Trees work by recursively splitting the data based on the features that result in the best separation (maximum information gain or Gini impurity reduction) of classes at each node.

**Key Aspects of Decision Trees:**

* **Nodes, Branches, and Leaves:**
  + **Root Node:** Represents the entire dataset.
  + **Internal Nodes:** Correspond to a feature and a decision rule, leading to subsequent nodes or leaves.
  + **Leaves (Terminal Nodes):** Represent the class label or target value.
* **Splitting Criteria:**
  + Decision Trees use various criteria (e.g., Gini impurity, information gain) to determine the best attribute to split the data at each node.
  + Gini impurity measures the probability of misclassifying an instance, while information gain measures the reduction in entropy after the split.
* **Decision Rules:**
  + Each internal node in the tree represents a decision based on a feature and a threshold.
  + Decision Trees learn these decision rules to partition the data into subsets that are more homogeneous in terms of the target variable.
* **Pruning:**
  + Techniques like pre-pruning (limiting the tree depth) or post-pruning (pruning the tree after it's built) are used to prevent overfitting and improve generalization.
* **Handling Categorical and Numerical Data:**
  + Decision Trees can handle both categorical and numerical features by selecting the optimal split points or categories.

**Workflow of Decision Tree Classifier:**

1. **Data Preprocessing:**
   * Handling missing values, encoding categorical variables, etc.
2. **Model Training:**
   * Building the decision tree by recursively partitioning the dataset based on feature splits that maximize information gain or minimize impurity.
3. **Tree Visualization:**
   * Visualizing the decision tree structure to understand the decision rules and feature importance.
4. **Pruning and Tuning:**
   * Pruning the tree to avoid overfitting and tuning hyperparameters like maximum depth, minimum samples per leaf, etc.
5. **Model Evaluation:**
   * Evaluating the model's performance using metrics like accuracy, precision, recall, F1-score, etc., on a test dataset.

**Advantages of Decision Tree Classifier:**

* Easy to understand and interpret. The decision rules are interpretable.
* Capable of handling both numerical and categorical data.
* Works well with large datasets.
* Can handle non-linear relationships in data.

**Limitations:**

* Prone to overfitting, especially with deep trees.
* Highly sensitive to noisy data and outliers.
* Can create biased trees if some classes dominate.

Decision Trees are fundamental models that serve as a basis for more complex ensemble methods like Random Forests, Gradient Boosting, etc. They are popular in various domains due to their simplicity, interpretability, and ability to handle a variety of data types. However, their tendency to overfit requires proper tuning and regularization techniques to build robust models.

**PROJECT INFERENCES**

**1)Logistic regression model classification report**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Not Fraud*** | *0.95* | *0.96* | *0.95* | *72* |
| **Fraud** | *0.96* | *0.94* | *0.95* | *70* |
| **Accuracy** |  |  | *0.95* | *142* |
| **Macro avg** | *0.95* | *0.95* | *0.95* | *142* |
| **Weighted avg** | *0.95* | *0.95* | *0.95* | *142* |

**Precision**   **Recall**  **F1-Score**  **Support**

The classification report provides various evaluation metrics for each class ('Not Fraud' and 'Fraud') and for the overall model performance on the validation set.

Here's an interpretation of the metrics in the classification report:

* **Precision:** Indicates the proportion of correctly identified instances of a class among all instances predicted as that class. For instance, the precision for 'Not Fraud' is 0.95, implying that 96% of the instances predicted as 'Not Fraud' were correctly classified.
* **Recall (Sensitivity):** Denotes the proportion of correctly identified instances of a class among all instances of that class in the dataset. For instance, the recall for 'Fraud' is 0.96, suggesting that 96% of the actual 'Fraud' instances were correctly identified.
* **F1-score:** The harmonic mean of precision and recall. It provides a balanced measure between precision and recall. For instance, the F1-score for both 'Fraud' and 'Not Fraud' is 0.95.
* **Support:** Indicates the number of actual occurrences of each class in the validation set ('Not Fraud' - 72 instances, 'Fraud' - 70 instances).
* **Accuracy:** Represents the ratio of correctly predicted instances to the total number of instances in the validation set. Here, the overall accuracy is 0.95, meaning 95% of instances were correctly classified.
* **Macro-average:** Calculates metrics independently for each class and then averages them. It doesn't consider class imbalance.
* **Weighted-average:** Similar to the macro-average, but it takes into account the number of instances for each class. This is weighted by the number of true instances in each class.

The classification report shows that the Logistic Regression model **performs well** on the validation set, achieving high precision and recall for both 'Not Fraud' and 'Fraud' classes. The F1-score, which is the harmonic mean of precision and recall, is also high for both classes. This suggests that the model is effective in classifying instances as 'Fraud' or 'Not Fraud' with good overall accuracy and balanced performance across both classes.

**2)Decision Tree Classifier Model classification Report**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Not Fraud*** | *0.93* | *0.92* | *0.92* | *72* |
| **Fraud** | *0.92* | *0.93* | *0.92* | *70* |
| **Accuracy** |  |  | *0.92* | *142* |
| **Macro avg** | *0.92* | *0.92* | *0.92* | *142* |
| **Weighted avg** | *0.92* | *0.20* | *0.92* | *142* |

**Precision**  **Recall**  F1-Score Support

Here's an interpretation of the classification report:

* **Precision:** Indicates the proportion of correctly identified instances of a class among all instances predicted as that class.
  + 'Not Fraud' precision is 0.93: Among instances predicted as 'Not Fraud', 95% were correctly classified.
  + 'Fraud' precision is 0.92: Among instances predicted as 'Fraud', 85% were correctly classified.
* **Recall (Sensitivity):** Denotes the proportion of correctly identified instances of a class among all instances of that class in the dataset.
  + 'Not Fraud' recall is 0.92: 92% of the actual 'Not Fraud' instances were correctly identified.
  + 'Fraud' recall is 0.93: 93% of the actual 'Fraud' instances were correctly identified.
* **F1-score:** The harmonic mean of precision and recall. It provides a balanced measure between precision and recall.
  + 'Not Fraud' F1-score is 0.92.
  + 'Fraud' F1-score is 0.92.
* **Support:** Indicates the number of actual occurrences of each class in the validation set ('Not Fraud' - 72 instances, 'Fraud' - 70 instances).
* **Accuracy:** Represents the ratio of correctly predicted instances to the total number of instances in the validation set. Here, the overall accuracy is 0.92, meaning 92% of instances were correctly classified.
* **Macro-average:** Calculates metrics independently for each class and then averages them. Here, the macro-averaged F1-score is 0.92.
* **Weighted-average:** Similar to the macro-average, but it takes into account the number of instances for each class. Here, the weighted-averaged F1-score is also 0.92.

Overall, the classification report provides an evaluation of the model's performance for both 'Not Fraud' and 'Fraud' classes, including precision, recall, and F1-score. The model demonstrates relatively high precision for both classes, but there's a notable difference in recall, especially for 'Not Fraud', where it's slightly lower. The F1-scores suggest a reasonably balanced performance between precision and recall for both classes.

3) **Linear Support Vector Classifier (LinearSVC) Model:**

Here's an interpretation of the metrics in the classification report:

* **Precision:** Indicates the accuracy of the positive predictions.
  + 'Not Fraud' precision is 0.95: Among instances predicted as 'Not Fraud', 95% were correctly classified.
  + 'Fraud' precision is 0.96: Among instances predicted as 'Fraud', 96% were correctly classified.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Not Fraud* | *0.95* | *0.96* | *0.95* | *72* |
| Fraud | *0.96* | *0.94* | *0.95* | *70* |
| Accuracy |  |  | *0.95* | *142* |
| Macro avg | *0.95* | *0.95* | *0.95* | *142* |
| Weighted avg | *0.95* | *0.95* | *0.95* | *142* |

Precision *Recall F1-Score Support*

* **Recall (Sensitivity):** Denotes the proportion of actual positives that were correctly identified.
  + 'Not Fraud' recall is 0.96: 96% of the actual 'Not Fraud' instances were correctly identified.
  + 'Fraud' recall is 0.94: 94% of the actual 'Fraud' instances were correctly identified.
* **F1-score:** The harmonic mean of precision and recall. It provides a balanced measure between precision and recall.
  + 'Not Fraud' F1-score is 0.95.
  + 'Fraud' F1-score is 0.95.
* **Support:** Indicates the number of actual occurrences of each class in the validation set ('Not Fraud' - 72 instances, 'Fraud' - 70 instances).
* **Accuracy:** Represents the ratio of correctly predicted instances to the total number of instances in the validation set. Here, the overall accuracy is 0.95, meaning 95% of instances were correctly classified.
* **Macro-average:** Calculates metrics independently for each class and then averages them. The macro-averaged F1-score is 0.95.
* **Weighted-average:** Similar to the macro-average, but it takes into account the number of instances for each class. The weighted-averaged F1-score is also 0.95.

Overall, the classification report indicates that the LinearSVC model with balanced class weights performs well. It shows high precision, recall, and F1-scores for both 'Not Fraud' and 'Fraud' classes. The model has an accuracy of 95%, demonstrating its effectiveness in distinguishing between 'Not Fraud' and 'Fraud' instances on the provided validation dataset.

**CONCLUSION**

Bottom of Form

Determining the "best" model depends on various factors and the specific context of the problem. The choice of the accurate model often involves considering multiple metrics, understanding the nature of the problem, and assessing the trade-offs between different evaluation criteria.

Let's recap the key evaluation metrics for the models:

1. **Decision Tree(DT) Model:**
   * Accuracy: 0.92
   * Recall (Fraud): 0.93
2. **Logistic Regression (LR) Model:**
   * Accuracy: 0.95
   * Recall (Fraud): 0.94
3. **Linear Support Vector Classifier (LinearSVC) Model:**
   * Accuracy: 0.95
   * Recall (Fraud): 0.94

* Based on the provided metrics and focusing on the detection of 'Fraud' cases (as indicated by recall for 'Fraud'), all three models have performed well in terms of accuracy, achieving scores ranging from 92% to 95%.
* Regarding the ability to identify 'Fraud' instances ('Fraud' class recall), LR and SVM models exhibit the same high recall score of 0.94,
* While the accuracy is an essential metric, especially for overall model performance, the high 'Fraud' class recall is crucial in fraud detection scenarios to minimize false negatives (i.e., correctly identifying fraudulent transactions).
* Based on the provided metrics alone, the Logistic Regression and Linear SVC models seem to have slightly higher accuracy and recall for the Fraud class.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Logistic Regression Model** | **Decision Tree Model** | **Support Vendor Classifier Model** |
| **Accuracy** | 0.95 | 0.92 | 0.95 |
| **F1-score-Fraud** | 0.95 | 0.92 | 0.95 |
| **Recall-Fraud** | 0.94 | 0.93 | 0.94 |