Titanic Survival Prediction Project Report

Introduction

This project involves predicting the survival of passengers on the Titanic using machine learning techniques. We implemented this using Python with libraries such as pandas, NumPy, and scikit-learn, and executed the project in both Jupyter Notebook and VS Code. Two models were trained: **Logistic Regression** and **Decision Tree Classifier**. The objective was to pre-process the dataset, train these models, evaluate their performance, and interpret the results.

Step 1: Loading the Dataset

We used the Titanic dataset available from a public GitHub repository. It contains 891 records with features like **PassengerId**, **Pclass**, **Name**, **Sex**, **Age**, **SibSp**, **Parch**, **Ticket**, **Fare**, **Cabin**, **Embarked**, and the target variable **Survived**.

We loaded the dataset using **pandas** and verified the first few entries to understand the data.

Step 2: Data Pre-processing

Actions Taken:

1. **Dropped Irrelevant Columns:**
   - `Name`, `Ticket`, and `Cabin` were removed since they do not contribute significantly to survival prediction or contain too many missing values.
2. **Handling Missing Values:**
   - `Age`: Missing values filled with the **median**.
   - `Embarked`: Filled with the **mode** (most frequent port).
3. **Encoding Categorical Variables:**
   - Converted `Sex` and `Embarked` into numeric form using **one-hot encoding**, enabling machine learning models to process these features.

After pre-processing, all data was numeric and ready for modeling.

Step 3: Feature and Target Selection

We separated the dataset into:

- **Features (X):** All columns except `Survived`.
- **Target (y):** The `Survived` column indicating whether the passenger survived (1) or not (0).

Step 4: Splitting the Data

We split the dataset into:

- **80% Training Data**
- **20% Test Data**

This was done using `train_test_split` from scikit-learn with a fixed random state for reproducibility. This allows evaluation of the model on unseen data.

Step 5: Logistic Regression Model

We trained a **Logistic Regression** model with a maximum iteration of 200. The model was trained on the training data and then used to predict outcomes on the test data.

Evaluation:

- **Accuracy:** ~80%
- **Classification Report:** Provided precision, recall, and F1-score for both survived and not-survived classes.
- **Confusion Matrix:** Showed the count of true positives, true negatives, false positives, and false negatives.

Observation:
Logistic Regression performed well with balanced precision and recall across both classes. This indicates the model's reliability in distinguishing between survivors and non-survivors.

Step 6: Decision Tree Classifier

We then trained a **Decision Tree Classifier** using the same training data.

Evaluation:

- **Accuracy:** ~72%
- **Classification Report:** Revealed lower precision and recall for the survivor class compared to logistic regression.
- **Confusion Matrix:** Demonstrated more misclassifications, particularly for survivors.

Observation:
While the Decision Tree is easy to interpret and visualize, its performance on this dataset was inferior to Logistic Regression. It was more prone to overfitting and less accurate on the test data.

Step 7: Model Comparison and Interpretation

Comparing both models:

- **Logistic Regression:** Higher accuracy and more balanced performance.
- **Decision Tree:** Lower accuracy and slightly more biased towards non-survivors.

Confusion Matrices:

- These matrices clearly showed that Logistic Regression made fewer misclassifications compared to the Decision Tree.

Classification Reports:

- These reports provided detailed insights into precision, recall, and F1-score, helping assess the strengths and weaknesses of each model.

Conclusion

Learning from this project:

- We cleaned and pre-processed the Titanic dataset.
- Built two predictive models: Logistic Regression and Decision Tree.
- Evaluated and compared both models using accuracy, classification reports, and confusion matrices.

Final Result:
The **Logistic Regression** model performed better on unseen data, making it a more suitable choice for this problem.