Project Report

Algorithm: Random Forest Classifier

Applying Data Preprocessing:

1. We first imported necessary libraries such as NumPy and pandas which we used for data preprocessing.

2. We imported the dataset into a Pandas dataframe using the function "pd.read_csv("train.csv")". We also created a backup file of this dataframe in order to have original copy of data.

3. After that, the data shape is printed, and missing values are checked by calling the ".isnull().sum()" function on the dataframe. The output shows that the Age, Cabin, and Embarked columns have missing values.

4. To handle these missing values, the following steps are taken:

    - For the Age column, the missing values are filled with the **mean** of the existing values using the ".fillna()" function.
    - For the Cabin column, the missing values are filled with the mode of the existing values using the ".fillna()" function.
    - For the Embarked column, the missing values are filled with the mode of the existing values using the ".fillna()" function.

5. **Summary:** Overall, the code performs the tasks of data cleansing by handling the missing values present in the dataset by replacing them with mean, mode respectively. The code is clear and efficient. We used comments in every line for better understanding of code.

Applying Feature Engineering:

1. In our code, One-hot Encoding is used as a technique of feature engineering.

2. We first converted the datatype of “Pclass” column from numerical to string by using the “.apply()” function. This is done because one-hot encoding is applied to all the categorical variables and “Pclass” column was numerical.

3. Then we used “.select_dtypes(include=['object']).columns” function to select all the columns having an object data type and saved them in a variable named “cat_cols”.

4. Then we used “.select_dtypes(include=['object']).columns” function to select all the columns having an object data type and saved them in a variable named “cat_cols”.

5. After that, we used the “pd.get_dummies ()” function to dummify the selected columns. This created a new set of binary columns, one for each unique category in the original column. We created new columns by using the original column's name as prefix and separating the column’s name by “_”.

6. **Summary**: One-hot encoding is widely used in machine learning to handle categorical variables and it's a powerful feature engineering technique. We applied feature engineering technique of one-hot encoding on the dataset and converted the categorical variables into numerical variables.

Applying the classification model:

1. We implemented a Random Forest classification algorithm to predict the survival of the passengers on the Titanic. The code consists of the following steps:

    - Removing the 'Survived' column from the dataframe and store it in the 'labels' variable. This column will serve as the target variable for the model.

    - We split the data into training and test sets using the train_test_split function from the sklearn.model_selection library. This function takes in three arguments, the data, labels, and the test size. In this case, the test size is set to 25%.

    - We then imported the RandomForestClassifier from the sklearn.ensemble library and created an object of random forest classifier.

    - We Fit the classifier on the training data and labels by calling the fit() function on the classifier object and passing the training data and labels as arguments.

    - Next, predictions are made on the test set by calling the predict_proba() function on the classifier object and passing the test set as an argument. The result is an array of probabilities for each sample.

    - To evaluate the performance of the model, we used ROC-AUC score. We imported the roc_auc_score() library from the sklearn.metrics for this purpose. This function takes in two arguments, the true labels (y_test) and the predicted probabilities (y_pred_proba).

    - Finally, the code prints the ROC-AUC score of the model.

Why we used Random Forest Classifier:

Random Forest is a powerful and widely used machine learning algorithm that is used for both classification and regression problems. In the case of the Titanic dataset, the goal is to predict whether a passenger survived or not, which is a classification problem. Random Forest is well-suited for this type of problem and can produce accurate predictions.

Elaboration:

Random Forest algorithm is an ensemble technique that creates a set of decision trees from a randomly selected subset of training data. Then, it averages the predictions of each decision tree to improve the overall accuracy of the model.

The algorithm is effective because it reduces overfitting, which is a common issue with decision tree models. In addition, it can handle large amounts of data and deal with missing values and categorical variables, which are present in the Titanic dataset.

The dataset contains a large number of features, some of them are categorical in nature, having many levels. The Random Forest algorithm is well-suited for this situation because it can handle categorical variables very well.

Justification:

Based on the characteristics of the Titanic dataset and the problem at hand, using a Random Forest algorithm is an appropriate choice. After evaluating the model using the ROC-AUC score, we found that the performance is high with a score of 0.87, which indicates that the model is able to make accurate predictions.