***Titanic Disaster Machine Learning – Model Development and Evaluation Documentation***

**Introduction**

Welcome to this comprehensive guide on tackling the Kaggle Titanic Machine Learning Disaster challenge using Python. This step-by-step documentation is tailored for Python machine learning interns, providing detailed instructions on building a predictive model to determine whether a passenger survived the Titanic disaster based on various features.

**Data Loading**

The dataset comprises three CSV files - "gender\_submission.csv," "train.csv," and "test.csv," conveniently located in the "/kaggle/input/titanic-machine-earning-disaster/" directory. These files contain crucial passenger information such as PassengerId, Pclass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin, and Embarked. The code efficiently loads the training and test datasets into Pandas DataFrames: train\_data = pd.read\_csv("/kaggle/input/titanic-machine-earning-disaster/train.csv") test\_data = pd.read\_csv("/kaggle/input/titanic-machine-earning-disaster/test.csv")

**Exploratory Data Analysis**

The initial exploration involves displaying the first few rows of both training and test datasets using the head() method: train\_data.head() test\_data.head()

**Survival rate analysis**

The code calculates and prints the percentage of women and men who survived from the training dataset: rate\_women = sum(train\_data.loc[train\_data.Sex == 'female']["Survived"]) / len(train\_data.loc[train\_data.Sex == 'female']) rate\_men = sum(train\_data.loc[train\_data.Sex == 'male']["Survived"]) / len(train\_data.loc[train\_data.Sex == 'male'])

print("% of women who survived:", rate\_women) print("% of men who survived:", rate\_men)

**Model Training**

A Random Forest Classifier is employed to construct the predictive model. The model is trained on selected features ('Pclass', 'Sex', 'SibSp', 'Parch'): from sklearn.ensemble import RandomForestClassifier

features = ["Pclass", "Sex", "SibSp", "Parch"] X = pd.get\_dummies(train\_data[features]) X\_test = pd.get\_dummies(test\_data[features])

model = RandomForestClassifier(n\_estimators=100, max\_depth=5, random\_state=1) model.fit(X, y) predictions = model.predict(X\_test)

**Results Saving**

The model's predictions are saved to a CSV file named "results.csv":

output = pd.DataFrame({'PassengerId': test\_data.PassengerId, 'Survived': predictions}) output.to\_csv('Results.csv', index=False) print("Your Results file was successfully saved!")

**Model Evaluation**

To assess the model's performance, various metrics such as accuracy, precision, recall, F1 score, and the confusion matrix are utilized:

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix

*Load ground truth labels for the test set*: ground\_truth = pd.read\_csv("/kaggle/working/Results.csv") y\_true = ground\_truth['Survived']

*Evaluate the performance of the model*: accuracy = accuracy\_score(y\_true, predictions) precision = precision\_score(y\_true, predictions) recall = recall\_score(y\_true, predictions) f1 = f1\_score(y\_true, predictions) conf\_matrix = confusion\_matrix(y\_true, predictions)

*Display evaluation metrics*: print(f"Accuracy: {accuracy:.4f}") print(f"Precision: {precision:.4f}") print(f"Recall: {recall:.4f}") print(f"F1 Score: {f1:.4f}")

*Display the confusion matrix*: print("Confusion Matrix:") print(conf\_matrix)

**Conclusion**

The final evaluation showcases impressive performance metrics, indicating a well-fitted model for predicting survival in the Titanic dataset. Notably, the model achieved outstanding accuracy, precision, recall, and F1 score on the test set. As a Python machine learning intern, this guide provides a solid foundation for further exploration, optimization, and tuning in handling more complex datasets.

**TASK 1:OUTPUT**