TASK 1

TITANIC SURVIVAL PREDICTION

Objective: Build a machine learning model using the famous Titanic dataset to predict whether a passenger survived or not based on features like age, gender, and class.

🔹 **Step 1: Import Libraries**

We start by importing the tools we need for data handling, visualization, and machine learning.

```
python
CopyEdit
import pandas as pd  # for data handling
import numpy as np  # for numerical operations
import seaborn as sns  # for data visualization
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,
classification_report
```

🔹 **Step 2: Load the Data**

```
python
CopyEdit
# Load data
```

```
train_data = pd.read_csv('train.csv')
test_data = pd.read_csv('test.csv')

# Let's look at the first few rows
train_data.head()
```

🔹 Step 3: Combine Data for Cleaning

We combine both datasets to apply the same cleaning steps consistently.

```
python
CopyEdit
# Add 'Survived' column to test data for uniformity
test_data['Survived'] = np.nan

# Combine datasets
combined_data = pd.concat([train_data, test_data],
sort=False)
```

🔹 Step 4: Clean and Prepare the Data

We clean missing values and convert text data (like gender) into numbers.

```
python
CopyEdit
# Fill missing Age values with median
combined_data['Age'].fillna(combined_data['Age'].median()
, inplace=True)

# Fill missing Fare values (only in test set)
```

```
combined_data['Fare'].fillna(combined_data['Fare'].median
(), inplace=True)

# Fill missing Embarked with the most common value
combined_data['Embarked'].fillna(combined_data['Embarked'
].mode()[0], inplace=True)

# Drop unnecessary columns
combined_data.drop(['Name', 'Ticket', 'Cabin'], axis=1,
inplace=True)

# Convert categorical columns into numbers
combined_data = pd.get_dummies(combined_data,
columns=['Sex', 'Embarked'], drop_first=True)
```

🔹 Step 5: Split the Data Back

Now we split the cleaned dataset back into training and testing parts.

```
python
CopyEdit
# Separate back into train and test sets
train_cleaned =
combined_data[combined_data['Survived'].notnull()]
test_cleaned =
combined_data[combined_data['Survived'].isnull()].drop('S
urvived', axis=1)

# Define input and target for training
X = train_cleaned.drop(['Survived', 'PassengerId'],
axis=1)
y = train_cleaned['Survived']
```

🔹 Step 6: Train the Model

We use a Random Forest Classifier – a powerful and simple algorithm – to train our model.

```
python
CopyEdit
# Split into training and validation sets
X_train, X_valid, y_train, y_valid = train_test_split(X,
y, test_size=0.2, random_state=42)

# Train the model
model = RandomForestClassifier(n_estimators=100,
random_state=42)
model.fit(X_train, y_train)

# Validate the model
y_pred = model.predict(X_valid)

# Print accuracy and report
print("Accuracy:", accuracy_score(y_valid, y_pred))
print(classification_report(y_valid, y_pred))
```

🔹 Step 7: Make Predictions and Prepare Submission File

Now we make predictions on the actual test data and create a CSV file for submission.

```
python
CopyEdit
# Prepare test features
X_test = test_cleaned.drop('PassengerId', axis=1)
```

```
# Make predictions
test_predictions = model.predict(X_test)

# Create submission file
submission = pd.DataFrame({
    'PassengerId':
test_cleaned['PassengerId'].astype(int),
    'Survived': test_predictions.astype(int)
})

# Save to CSV
submission.to_csv('submission.csv', index=False)
print("Submission file created!")
```

🔹 Step 8: (Optional) Check Feature Importance

Let’s see which features were most important for our model.

```
python
CopyEdit
importances = model.feature importances
features = X.columns
indices = np.argsort(importances)[::-1]

plt.figure(figsize=(10, 6))
plt.title("Important Features")
plt.bar(range(len(importances)), importances[indices],
color='skyblue')
plt.xticks(range(len(importances)), [features[i] for i in
indices], rotation=90)
plt.tight_layout()
plt.show()
```