Task 1

Titanic classification :-

Goal of the Code

This code predicts whether a person **survived or not** on the Titanic based on different factors like:

- Passenger class (Pclass)
- Gender (Sex)
- Age
- Number of siblings/spouses on board (SibSp)
- Number of parents/children on board (Parch)
- Ticket Fare (Fare)
- Boarding location (Embarked)

We use **Machine Learning (Random Forest Classifier)** to make predictions.

Follow this steps to make code :-

1. Importing Required Libraries

```
python
CopyEdit
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
```

- **pandas** → Works with structured data (tables)
- **numpy** → Handles numerical data efficiently
- **seaborn & matplotlib** → Used for data visualization
- **sklearn (scikit-learn)** → Used for machine learning tasks

2. Load Titanic Dataset

python
CopyEdit
data =
pd.read_csv("https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv")

- Loads the **Titanic dataset** from an online source into a Pandas DataFrame.
- The dataset contains details about passengers (name, age, gender, ticket class, survival status, etc.).

3. Selecting Important Features

python
CopyEdit
features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']
data = data[features + ['Survived']]

- We **select only relevant columns** (features) that might affect survival.
- The Survived column is the **output variable** (0 = Did not survive, 1 = Survived).

4. Handling Missing Values

python
CopyEdit
data['Age'].fillna(data['Age'].median(), inplace=True)
data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)

- **Missing Age values** are filled with the **median** age of all passengers.
- **Missing Embarked (boarding location) values** are filled with the most common value (mode).

5. Converting Categorical Data to Numbers

```
python
CopyEdit
le = LabelEncoder()
data['Sex'] = le.fit_transform(data['Sex'])
data['Embarked'] = le.fit_transform(data['Embarked'])
```

- **Sex (Male/Female) and Embarked (C/Q/S)** are categorical variables (text), so we convert them into numbers using LabelEncoder().
    - **Male → 1, Female → 0**
    - **Embarked (C = 0, Q = 1, S = 2)**

6. Splitting Data into Training & Testing Sets

```
python
CopyEdit
X = data[features]
y = data['Survived']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

- **X** → Features (Pclass, Sex, Age, etc.)
- **y** → Target (Survived or Not)
- train_test_split() splits data into:
    - **80% Training Data** → Used to train the model
    - **20% Testing Data** → Used to test the model

7. Train the Model using Random Forest

```
python
CopyEdit
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

- **RandomForestClassifier** is a machine learning algorithm that creates multiple decision trees to make predictions.
- We **train the model** using the training data (X_train and y_train).

8. Make Predictions

python
CopyEdit
y_pred = model.predict(X_test)

- The trained model **predicts survival** on test data (X_test).
- y_pred stores the predictions (0 or 1).

9. Check Model Accuracy

python
CopyEdit
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
print(classification_report(y_test, y_pred))

- **Accuracy Score** tells how well the model predicts survival.
- **Classification Report** provides:
  - **Precision** (How many predicted survivors were correct?)
  - **Recall** (How many actual survivors were correctly predicted?)

10. Feature Importance (Which Factors Were Most Important?)

python
CopyEdit
importance = model.feature_importances_
feature_importance = pd.DataFrame({'Feature': features, 'Importance': importance}).sort_values(by='Importance', ascending=False)

```
print(feature_importance)
```

- **Feature Importance** tells which features contributed most to survival.

11. Visualizing Feature Importance

```
python
CopyEdit
sns.barplot(x=feature_importance['Importance'], y=feature_importance['Feature'])
plt.title("Feature Importance")
plt.show()
```

- This creates a **bar chart** to show the most important factors for survival.

Final Output

1. **Model Accuracy** (Example: Accuracy: 0.82 → 82% correct predictions)
2. **Classification Report** (Shows model performance for each class)
3. **Feature Importance Graph** (Shows which factors affected survival the most)

Summary

✅ This code **analyzes Titanic passenger data** and predicts survival using machine learning.
✅ It considers **age, gender, class, fare, and family relationships** to make predictions.
✅ We used **RandomForestClassifier**, which is a strong and accurate algorithm.