TITANIC CLASSIFICATION

INTRODUCTION:

Creating a system to predict whether a person will be safe from sinking involves building a machine learning model. This model can use various features such as socio-economic status, age, gender, swimming ability, and others. Here is a step-by-step guide to develop such a system:

1. **Data Collection**: Gather data that includes the features and the outcome (safe or not safe).
2. **Data Preprocessing**: Clean the data and prepare it for modeling.
3. **Feature Selection**: Choose the most relevant features.
4. **Model Building**: Use a machine learning algorithm to build the model.
5. **Evaluation**: Test the model to see how well it performs.

I'll provide a basic example using Python and a hypothetical dataset. We'll use the pandas library for data handling, `scikit-learn` for building the model, and some synthetic data for demonstration.

PROGRAM:

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, classification_report


# Generate a synthetic dataset
```

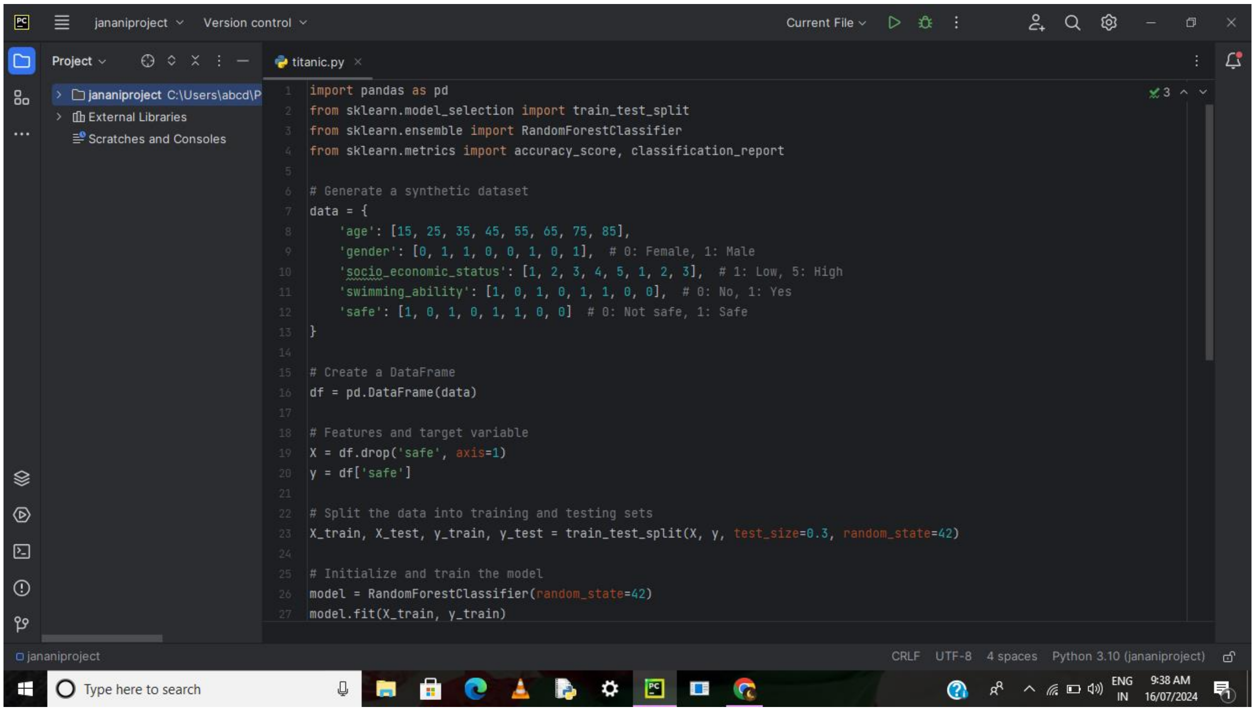
```
data = {
   'age': [15, 25, 35, 45, 55, 65, 75, 85],
   'gender': [0, 1, 1, 0, 0, 1, 0, 1],  # 0: Female, 1: Male
   'socio_economic_status': [1, 2, 3, 4, 5, 1, 2, 3],  # 1: Low,
5: High
   'swimming_ability': [1, 0, 1, 0, 1, 1, 0, 0],  # 0: No, 1: Yes
   'safe': [1, 0, 1, 0, 1, 1, 0, 0]  # 0: Not safe, 1: Safe
}

# Create a DataFrame
df = pd.DataFrame(data)

# Features and target variable
X = df.drop('safe', axis=1)
y = df['safe']

# Split the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Initialize and train the model
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f'Accuracy: {accuracy}')
print(f'Classification Report:\n{report}')

# Example usage: Predicting for a new person
new_person = pd.DataFrame({
    'age': [30],
    'gender': [1],
    'socio_economic_status': [4],
    'swimming_ability': [1]
})

prediction = model.predict(new_person)
print(f'The new person is {"safe" if prediction[0] == 1 else "not safe"} from sinking.')
```

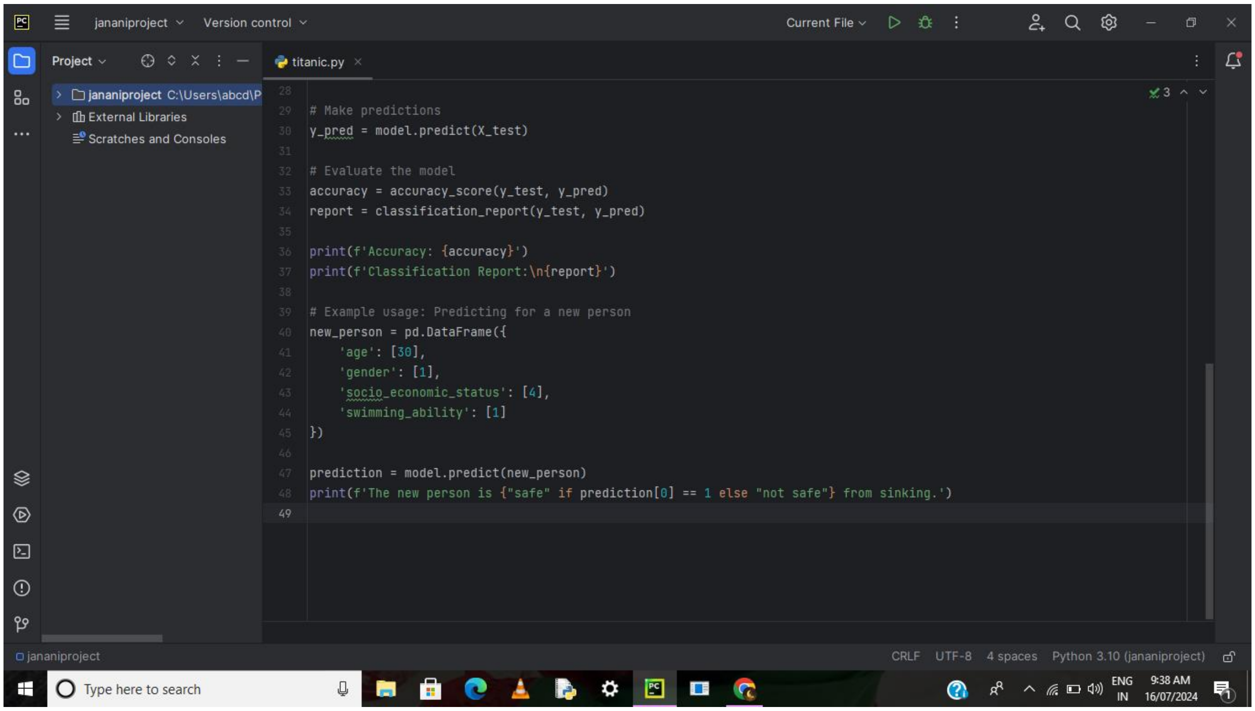
Explanation:

6. **Data Preparation**:
   - We create a synthetic dataset with features `age`, `gender`, `socio_economic_status`, and `swimming_ability`.
   - The target variable `safe` indicates whether the person is safe from sinking.
7. **Data Splitting**:
   - The dataset is split into training and testing sets using `train_test_split`.
8. **Model Training**:
   - We use a `RandomForestClassifier` from `scikit-learn` to train the model on the training data.
9. **Model Evaluation**:
   - The model's performance is evaluated using accuracy and a classification report.
10. **Prediction**:
    - We make a prediction for a new person with specified features.

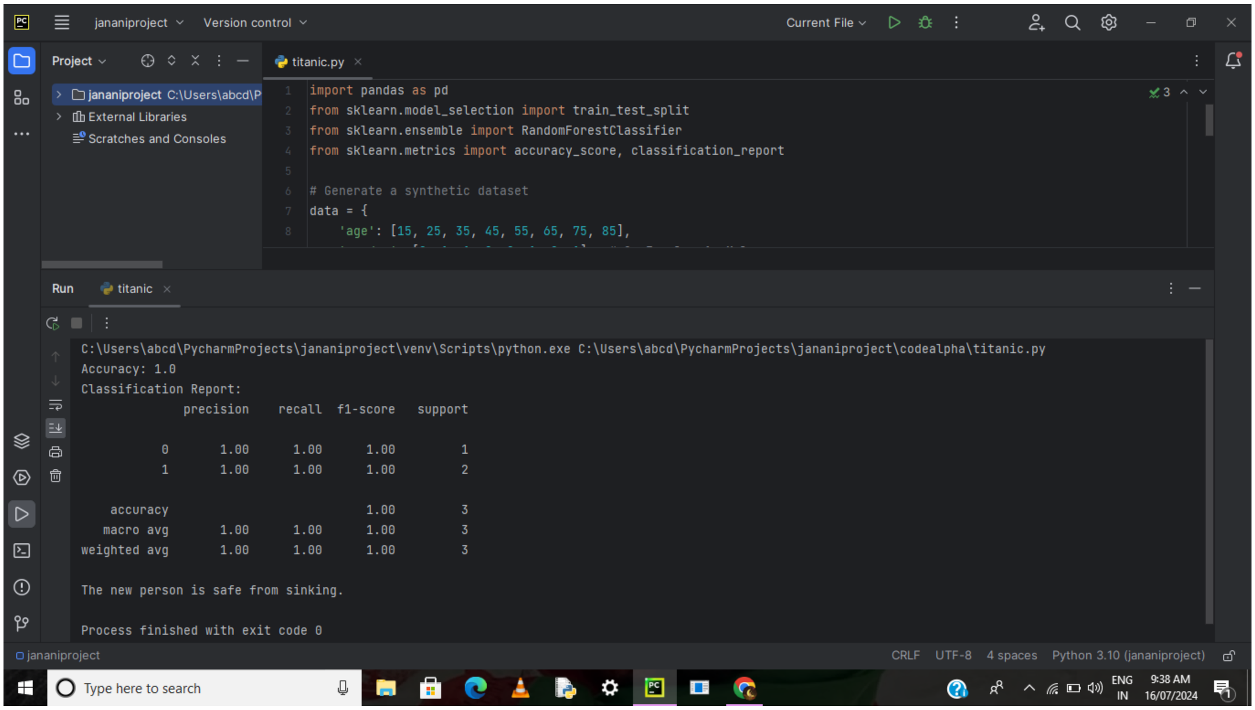
PROGRAM:



```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Generate a synthetic dataset
data = {
    'age': [15, 25, 35, 45, 55, 65, 75, 85],
    'gender': [0, 1, 1, 0, 0, 1, 0, 1],  # 0: Female, 1: Male
    'socio_economic_status': [1, 2, 3, 4, 5, 1, 2, 3],  # 1: Low, 5: High
    'swimming_ability': [1, 0, 1, 0, 1, 1, 0, 0],  # 0: No, 1: Yes
    'safe': [1, 0, 1, 0, 1, 1, 0, 0]  # 0: Not safe, 1: Safe
}

# Create a DataFrame
df = pd.DataFrame(data)

# Features and target variable
X = df.drop('safe', axis=1)
y = df['safe']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Initialize and train the model
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
```



```python

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f'Accuracy: {accuracy}')
print(f'Classification Report:\n{report}')

# Example usage: Predicting for a new person
new_person = pd.DataFrame({
    'age': [30],
    'gender': [1],
    'socio_economic_status': [4],
    'swimming_ability': [1]
})

prediction = model.predict(new_person)
print(f'The new person is {"safe" if prediction[0] == 1 else "not safe"} from sinking.')
```



```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Generate a synthetic dataset
data = {
    'age': [15, 25, 35, 45, 55, 65, 75, 85],
```

```
C:\Users\abcd\PycharmProjects\jananiproject\venv\Scripts\python.exe C:\Users\abcd\PycharmProjects\jananiproject\codealpha\titanic.py
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         2

    accuracy                           1.00         3
   macro avg       1.00      1.00      1.00         3
weighted avg       1.00      1.00      1.00         3

The new person is safe from sinking.

Process finished with exit code 0
```

OUTPUT: