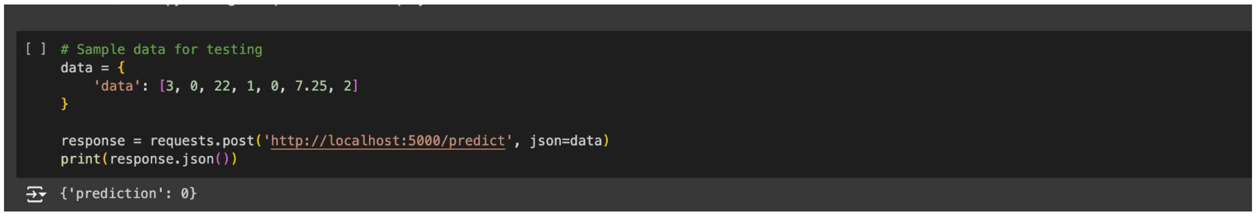
Contains -

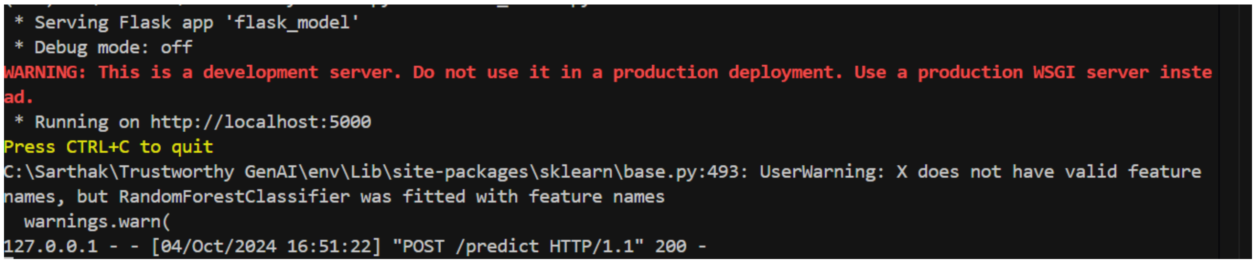
- **Data Selection and Preparation: (week_4_assignment.ipynb)**

1. Model Training:
2. Model Saving:
3. Flask App Creation:
4. Running the Flask App:
5. Testing the Deployed Model:

   - a sample API request and response
   - 

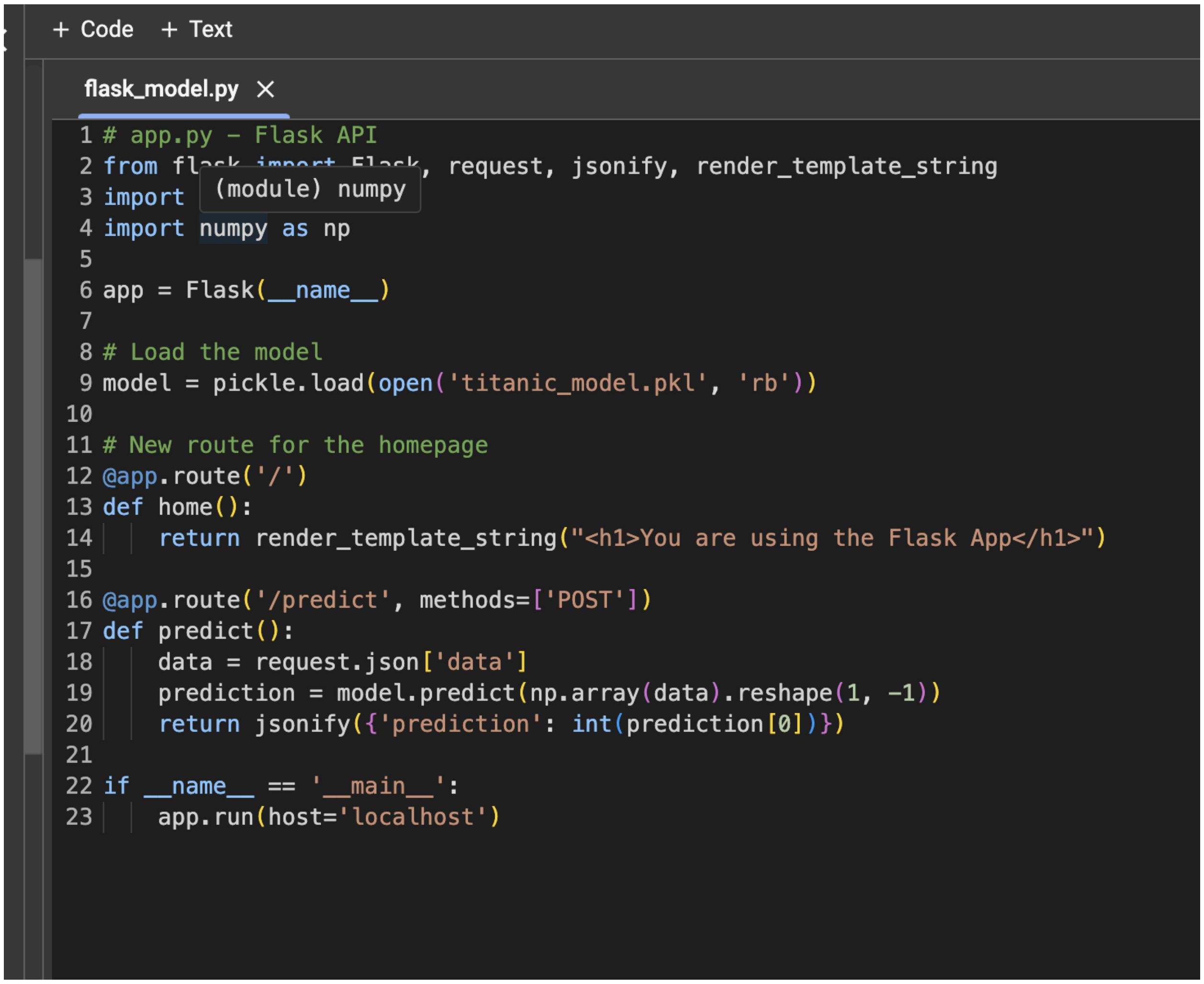
```
[ ] # Sample data for testing
    data = {
        'data': [3, 0, 22, 1, 0, 7.25, 2]
    }

    response = requests.post('http://localhost:5000/predict', json=data)
    print(response.json())

    {'prediction': 0}
```

Flask App running



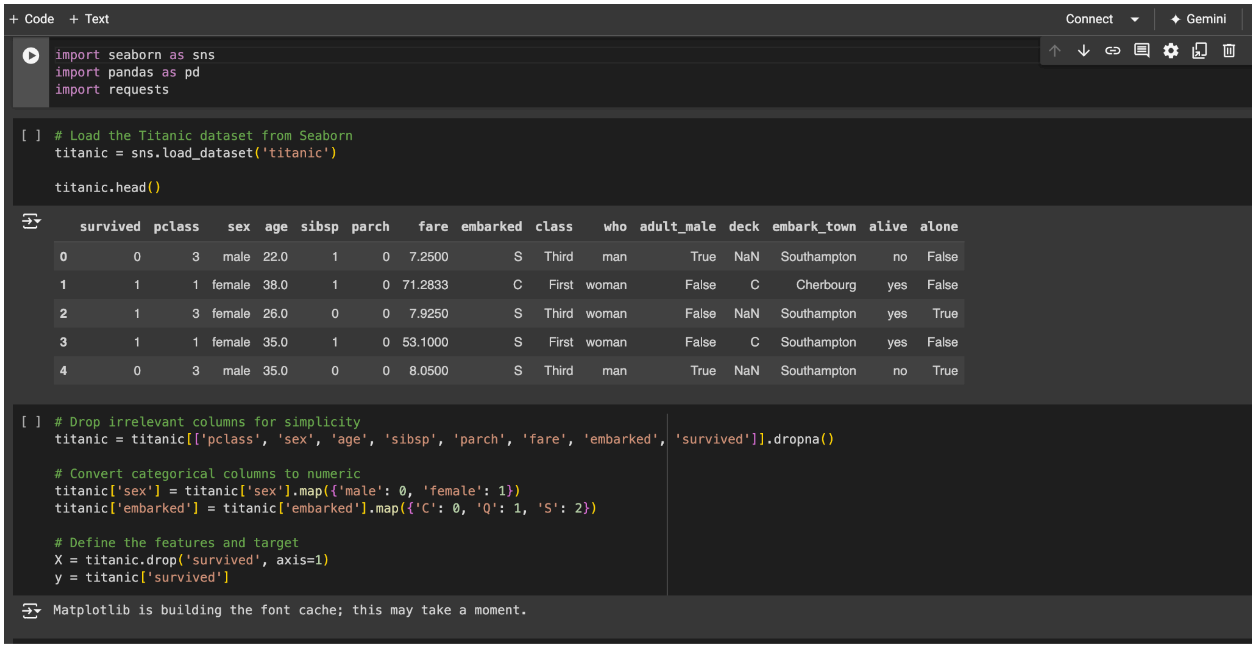
```
 * Serving Flask app 'flask_model'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server inste
ad.
 * Running on http://localhost:5000
Press CTRL+C to quit
C:\Sarthak\Trustworthy GenAI\env\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature
names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
127.0.0.1 - - [04/Oct/2024 16:51:22] "POST /predict HTTP/1.1" 200 -
```

   - Screenshot of the Flask app code



```python
# app.py - Flask API
from flask import Flask, request, jsonify, render_template_string
import
import numpy as np

app = Flask(__name__)

# Load the model
model = pickle.load(open('titanic_model.pkl', 'rb'))

# New route for the homepage
@app.route('/')
def home():
    return render_template_string("<h1>You are using the Flask App</h1>")

@app.route('/predict', methods=['POST'])
def predict():
    data = request.json['data']
    prediction = model.predict(np.array(data).reshape(1, -1))
    return jsonify({'prediction': int(prediction[0])})

if __name__ == '__main__':
    app.run(host='localhost')
```

○ Screenshot of the model creation and training process



```
import seaborn as sns
import pandas as pd
import requests
```

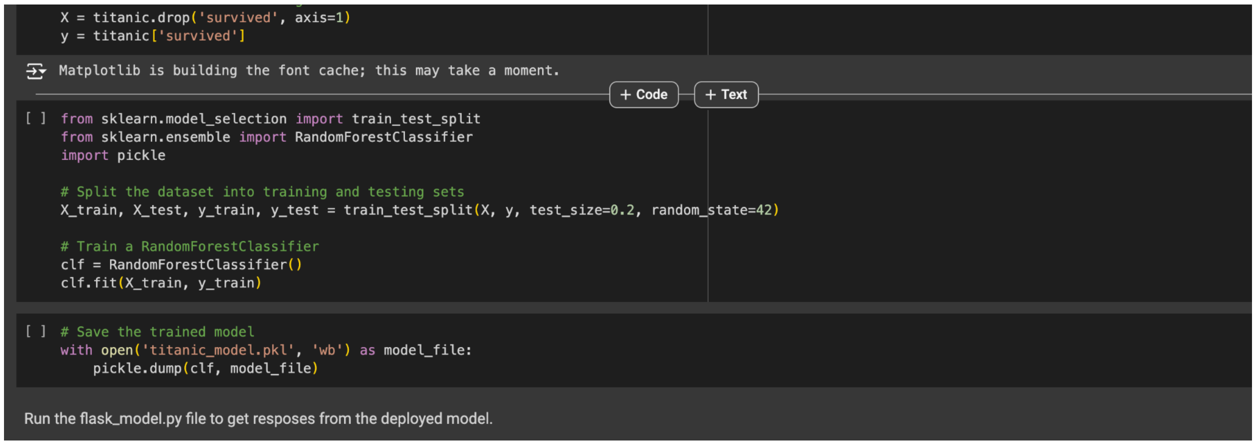
```
# Load the Titanic dataset from Seaborn
titanic = sns.load_dataset('titanic')

titanic.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

```
# Drop irrelevant columns for simplicity
titanic = titanic[['pclass', 'sex', 'age', 'sibsp', 'parch', 'fare', 'embarked', 'survived']].dropna()

# Convert categorical columns to numeric
titanic['sex'] = titanic['sex'].map({'male': 0, 'female': 1})
titanic['embarked'] = titanic['embarked'].map({'C': 0, 'Q': 1, 'S': 2})

# Define the features and target
X = titanic.drop('survived', axis=1)
y = titanic['survived']
```

```
Matplotlib is building the font cache; this may take a moment.
```

○



```
X = titanic.drop('survived', axis=1)
y = titanic['survived']
```

```
Matplotlib is building the font cache; this may take a moment.
```

```
[ ]  from sklearn.model_selection import train_test_split
     from sklearn.ensemble import RandomForestClassifier
     import pickle

     # Split the dataset into training and testing sets
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

     # Train a RandomForestClassifier
     clf = RandomForestClassifier()
     clf.fit(X_train, y_train)
```

```
[ ]  # Save the trained model
     with open('titanic_model.pkl', 'wb') as model_file:
         pickle.dump(clf, model_file)
```

Run the flask_model.py file to get resposes from the deployed model.

titanic_model.pkl - model pickle file

flask_model.py - python file for flask application