Titanic Survival Prediction Model Deployment with Flask

Name: Deployment on Flask (Titanic Survival Prediction Model Deployment with Flask)

Report date: 2025.02.26

Internship Batch: LISUM42

Version: 1.0

Data intake by: Abdukhakimov Asatilla

Data intake reviewer: Data Glacier Team

Titanic Tabular data details:

Total number of observations	891
Total number of files	1
Total number of features	12
Base format of the file	.csv
Size of the data	83.7KB

Dataset Selection

The dataset chosen for this task is the Titanic dataset, which contains passenger details and survival status. The goal is to build a model that predicts whether a passenger survived or not based on their Pclass, Gender, Age, and Fare.

Import Libraries

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import numpy as np
from flask import Flask, request, jsonify
import pickle

import warnings
warnings.filterwarnings("ignore")
```

Understanding Data

```
url = "https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv"
df = pd.read_csv(url)
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
df.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Let's preprocess data by handling missing values and encoding categorical variables.

```
df = df[['Pclass', 'Sex', 'Age', 'Fare', 'Survived']].dropna()
df['Sex'] = df['Sex'].map({'male': 0, 'female': 1})
```

```
df.head()
```

	Pclass	Sex	Age	Fare	Survived
0	3	0	22.0	7.2500	0
1	1	1	38.0	71.2833	1
2	3	1	26.0	7.9250	1
3	1	1	35.0	53.1000	1
4	3	0	35.0	8.0500	0

Model Training

Let's select features and target

```
X = df[['Pclass', 'Sex', 'Age', 'Fare']]
y = df['Survived']
```

```
# Split data into test and train
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

To train our model, we will use Random Forest Classifier

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

```
RandomForestClassifier
RandomForestClassifier()
```

```
# Save model
with open("titanic_model.pkl", "wb") as f:
    pickle.dump(model, f)
```

Deploying Model with Flask

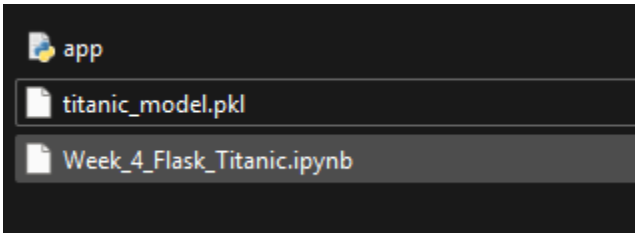
First, we load the trained model

```
# Load the Titanic model
with open("titanic_model.pkl", "rb") as f:
    model = pickle.load(f)
```

```
app = Flask(__name__)

@app.route('/')
def home():
    return "Titanic Survival Prediction API is running!"

@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json()  # Get JSON data from request

    # Extract features from the incoming JSON data
    pclass = data.get("Pclass")
    sex = data.get("Sex")
    age = data.get("Age")
    fare = data.get("Fare")

    # Convert categorical variable (Sex) to numerical
    sex_numeric = 1 if sex.lower() == 'male' else 0

    # Prepare the features array
    features = np.array([[pclass, sex_numeric, age, fare]])

    # Make prediction
    prediction = model.predict(features)

    # Return prediction result as JSON
    return jsonify({"survived": int(prediction[0])})
```

```
if __name__ == '__main__':
    app.run(use_reloader=False, port=5000)  # Run on port 5000
```

```
 * Serving Flask app '__main__'
 * Debug mode: off
```




```
C:\Users\treme\Data_glacier_w4\new>curl -X POST http://127.0.0.1:5000/predict -H "Content-Type:
application/json" -d "{\"Pclass\": 1, \"Sex\": \"male\", \"Age\": 22, \"Fare\": 7.25}"
{"survived":1}

C:\Users\treme\Data_glacier_w4\new>curl -X POST http://127.0.0.1:5000/predict -H "Content-Type:
application/json" -d "{\"Pclass\": 3, \"Sex\": \"male\", \"Age\": 40, \"Fare\": 7}"
{"survived":0}
```

The Flask API that predicts whether a passenger survived based on Pclass, Sex, Age, and Fare has been successfully created and working well.

- For a male passenger in **1st class** who is **22 years old** and paid **$7.25**, the prediction returned {"survived": 1} (indicating survival).
- For a male passenger in **3rd class** who is **40 years old** and paid **$7**, the prediction returned {"survived": 0} (indicating no survival).