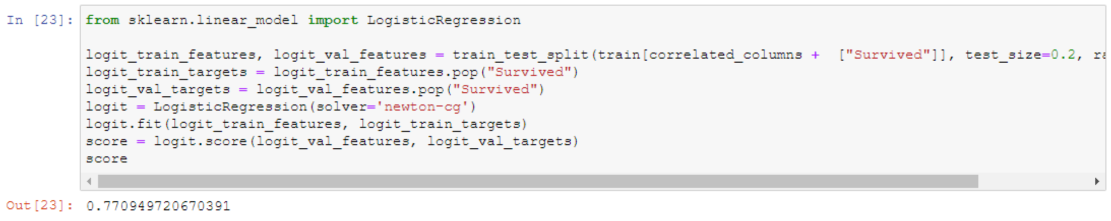
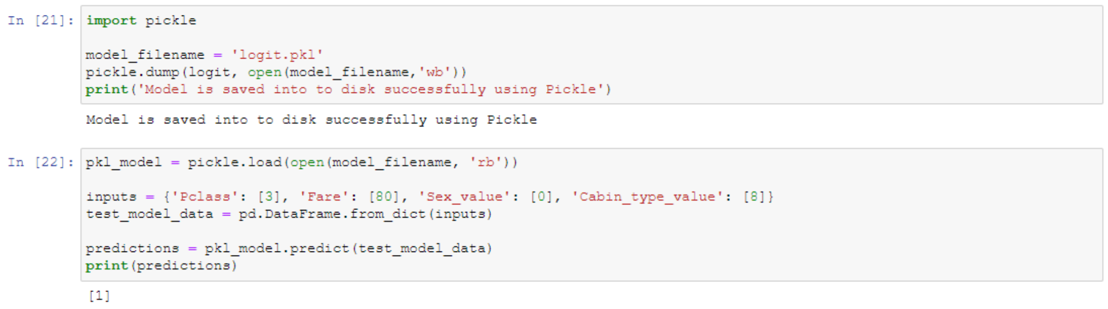
Name: Giorgos Moysiadis
Internship Batch Code: LISUM02
Submission Data: 15/8/2021
Submitted to: Data Glacier

After having solved and fixed the missing values and the categorical data (which can be seen in the eda notebook for more details) in the Titanic dataset I set up my model.



```
In [23]: from sklearn.linear_model import LogisticRegression

         logit_train_features, logit_val_features = train_test_split(train[correlated_columns +  ["Survived"]], test_size=0.2, r
         logit_train_targets = logit_train_features.pop("Survived")
         logit_val_targets = logit_val_features.pop("Survived")
         logit = LogisticRegression(solver='newton-cg')
         logit.fit(logit_train_features, logit_train_targets)
         score = logit.score(logit_val_features, logit_val_targets)
         score

Out[23]: 0.770949720670391
```

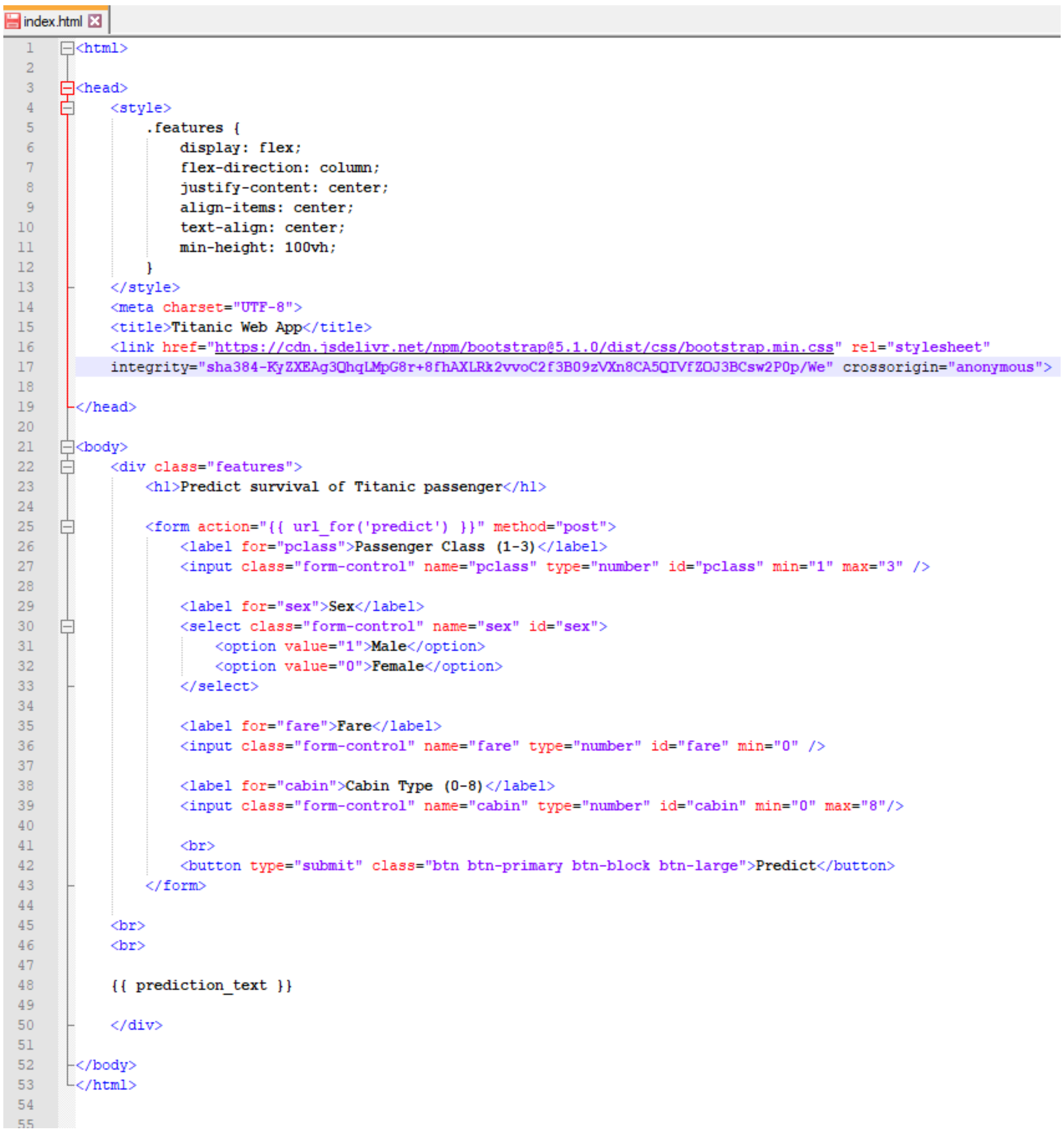
After that I import pickle for saving the model, as it can be seen in the next screenshot.



```
In [21]: import pickle

         model_filename = 'logit.pkl'
         pickle.dump(logit, open(model_filename,'wb'))
         print('Model is saved into to disk successfully using Pickle')

         Model is saved into to disk successfully using Pickle

In [22]: pkl_model = pickle.load(open(model_filename, 'rb'))

         inputs = {'Pclass': [3], 'Fare': [80], 'Sex_value': [0], 'Cabin_type_value': [8]}
         test_model_data = pd.DataFrame.from_dict(inputs)

         predictions = pkl_model.predict(test_model_data)
         print(predictions)

         [1]
```

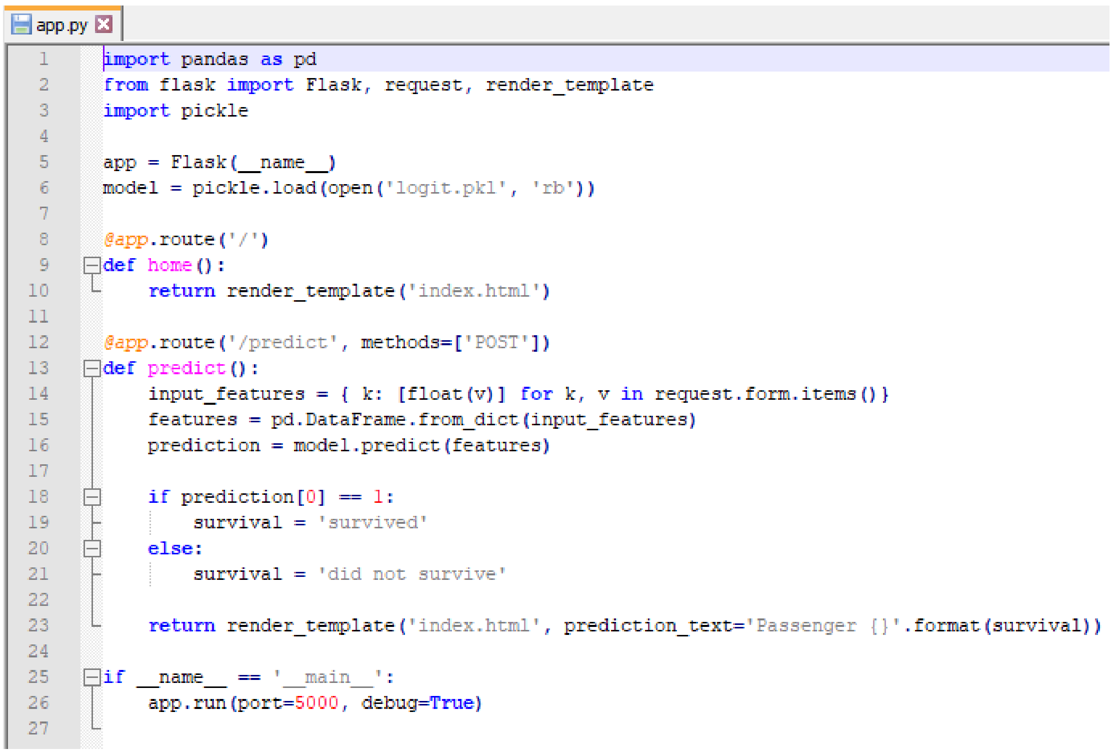
The model is saved successfully and loaded again in order to make sure it is working.

After saving the model I am creating a separate folder in order to put the app.py code, the saved pickle file for the model and another folder named “template” in order to store the index.html page.
In the next screenshot is the set up for the index.html page where my website will be working on.



```html
<html>

<head>
    <style>
        .features {
            display: flex;
            flex-direction: column;
            justify-content: center;
            align-items: center;
            text-align: center;
            min-height: 100vh;
        }
    </style>
    <meta charset="UTF-8">
    <title>Titanic Web App</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/css/bootstrap.min.css" rel="stylesheet"
    integrity="sha384-KyZXEAg3QhqLMpG8r+8fhAXLRk2vvoC2f3B09zVXn8CA5QIVfZOJ3BCsw2P0p/We" crossorigin="anonymous">

</head>

<body>
    <div class="features">
        <h1>Predict survival of Titanic passenger</h1>

        <form action="{{ url_for('predict') }}" method="post">
            <label for="pclass">Passenger Class (1-3)</label>
            <input class="form-control" name="pclass" type="number" id="pclass" min="1" max="3" />

            <label for="sex">Sex</label>
            <select class="form-control" name="sex" id="sex">
                <option value="1">Male</option>
                <option value="0">Female</option>
            </select>

            <label for="fare">Fare</label>
            <input class="form-control" name="fare" type="number" id="fare" min="0" />

            <label for="cabin">Cabin Type (0-8)</label>
            <input class="form-control" name="cabin" type="number" id="cabin" min="0" max="8"/>

            <br>
            <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
        </form>

    <br>
    <br>

    {{ prediction_text }}

    </div>

</body>
</html>
```

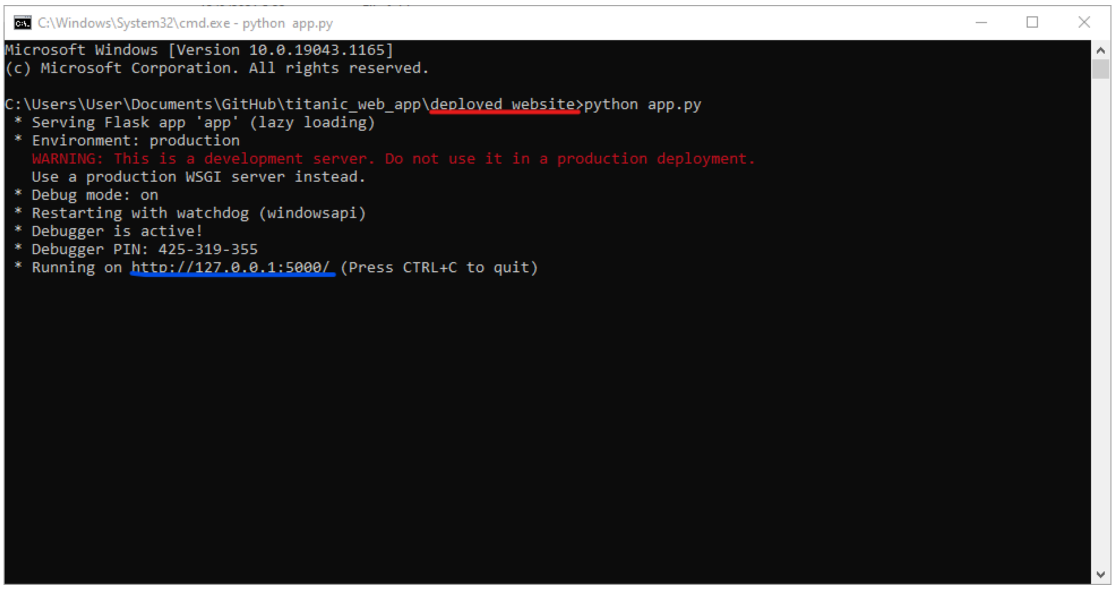
The next screenshot shows the app.py file where I'm loading the model from the pickle file.
Next I'm requesting the data, from the form that I have created in the index page, and they are fed then into the model to make the prediction and send the message in the "prediction_text" that I have set up in the index page.



app.py

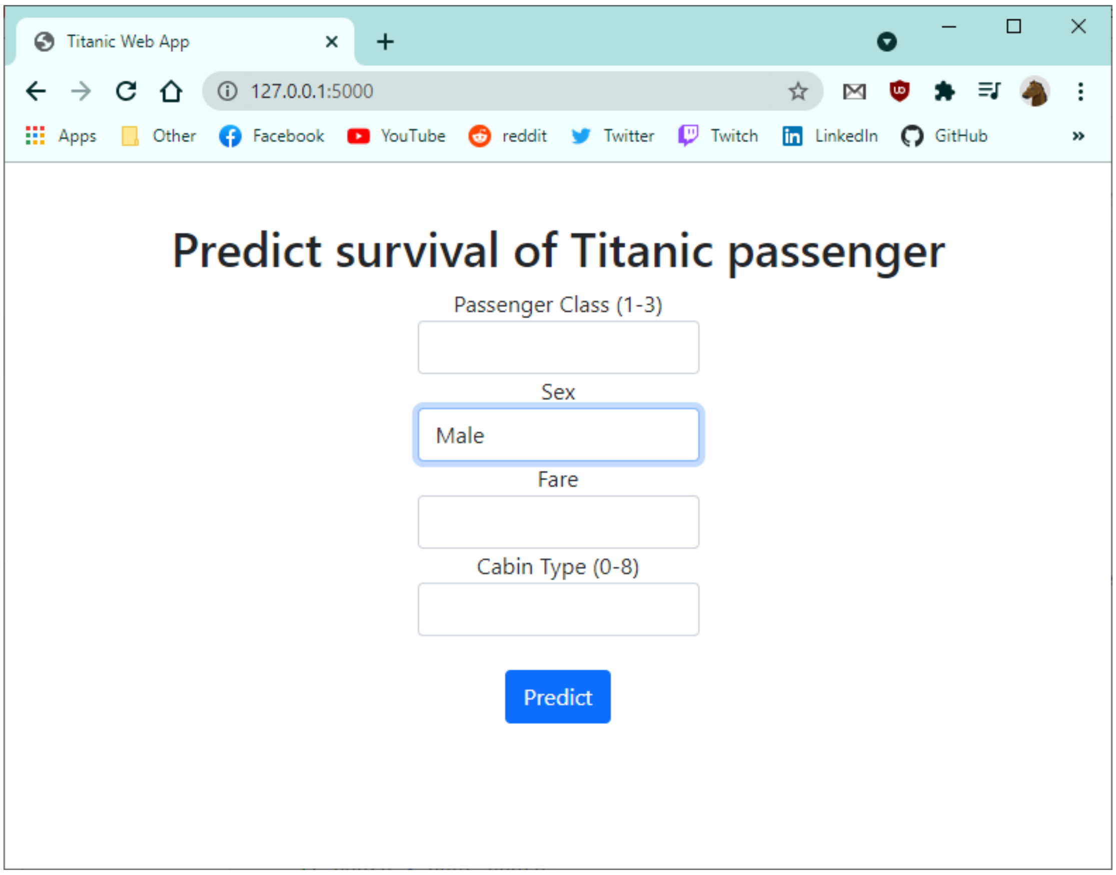
```
import pandas as pd
from flask import Flask, request, render_template
import pickle

app = Flask(__name__)
model = pickle.load(open('logit.pkl', 'rb'))

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    input_features = { k:  [float(v)] for k, v in request.form.items()}
    features = pd.DataFrame.from_dict(input_features)
    prediction = model.predict(features)

    if prediction[0] == 1:
        survival = 'survived'
    else:
        survival = 'did not survive'

    return render_template('index.html', prediction_text='Passenger {}'.format(survival))

if __name__ == '__main__':
    app.run(port=5000, debug=True)
```

After that, I am running the command prompt in the appropriate folder where my app.py file is.



```
C:\Windows\System32\cmd.exe - python  app.py
Microsoft Windows [Version 10.0.19043.1165]
(c) Microsoft Corporation. All rights reserved.

C:\Users\User\Documents\GitHub\titanic_web_app\deployed website>python app.py
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 425-319-355
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

I load the page in the blue underlined IP address and we have the next screenshot.



Predict survival of Titanic passenger

Passenger Class (1-3)

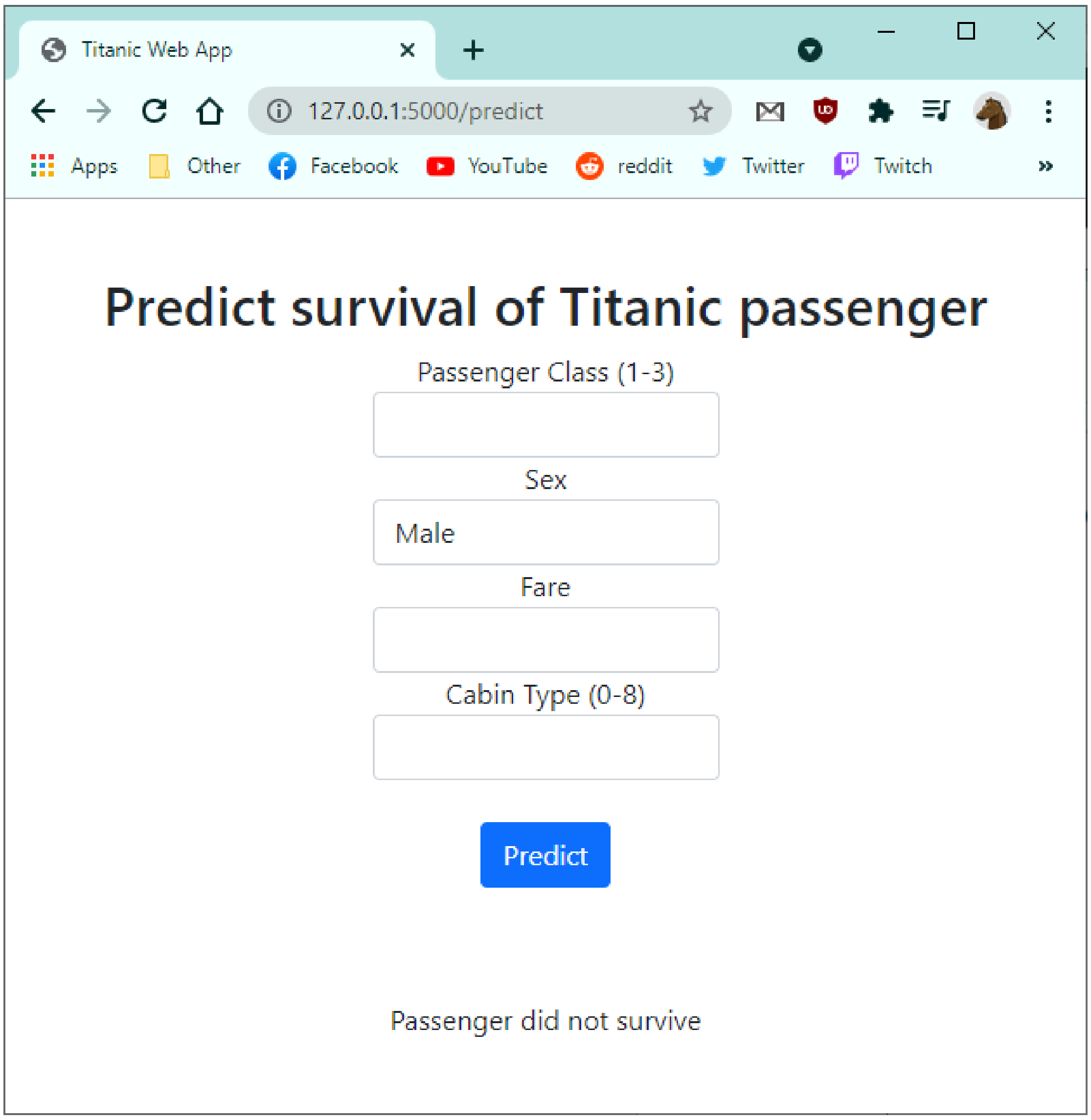
Sex

Male

Fare

Cabin Type (0-8)

We input some “random” values into the text fields such as Passenger Class = 2, Sex = Female, Fare = 20 and Cabin Type = 2.
After we hit predict we have the following screenshot with the result of the survival of our passenger.



Predict survival of Titanic passenger

Passenger Class (1-3)

Sex

Male

Fare

Cabin Type (0-8)

Passenger did not survive

And as it can be seen the url changes once the “Predict” button is pushed and our prediction is printed.