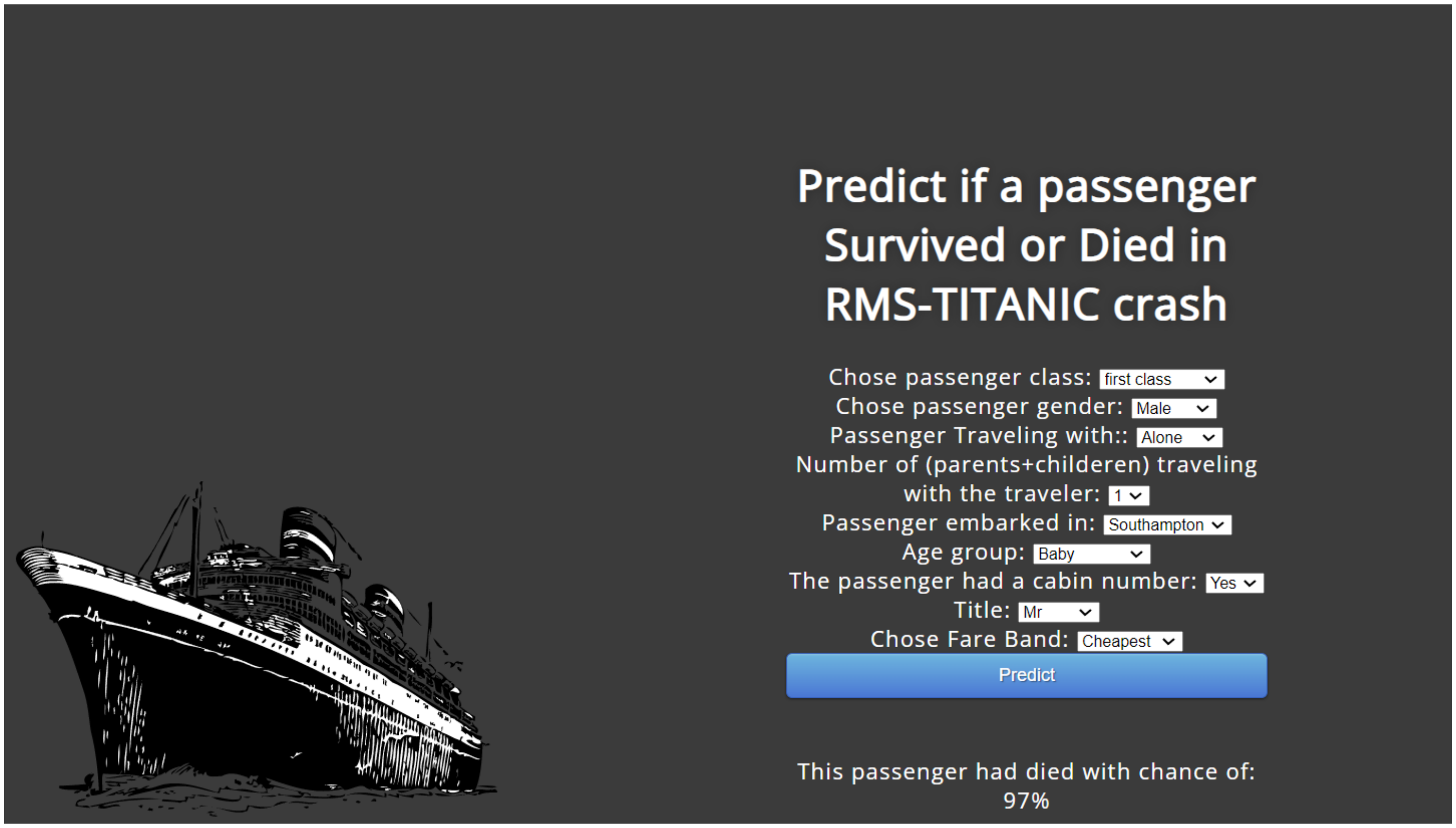
Name: Alireza Samadifardheris   Batch code=LISUM14

Submission date: 27/10/2022   Submitted to Canvas, Github

This Application which is deployed on flask is a model trained over the passengers' dataset of RMS Titanic and if they have survived or not. The aim is to have a web application based on the pickled-trained model to predict if a passenger could have survived or died. Final application would look like the following:



Predict if a passenger Survived or Died in RMS-TITANIC crash

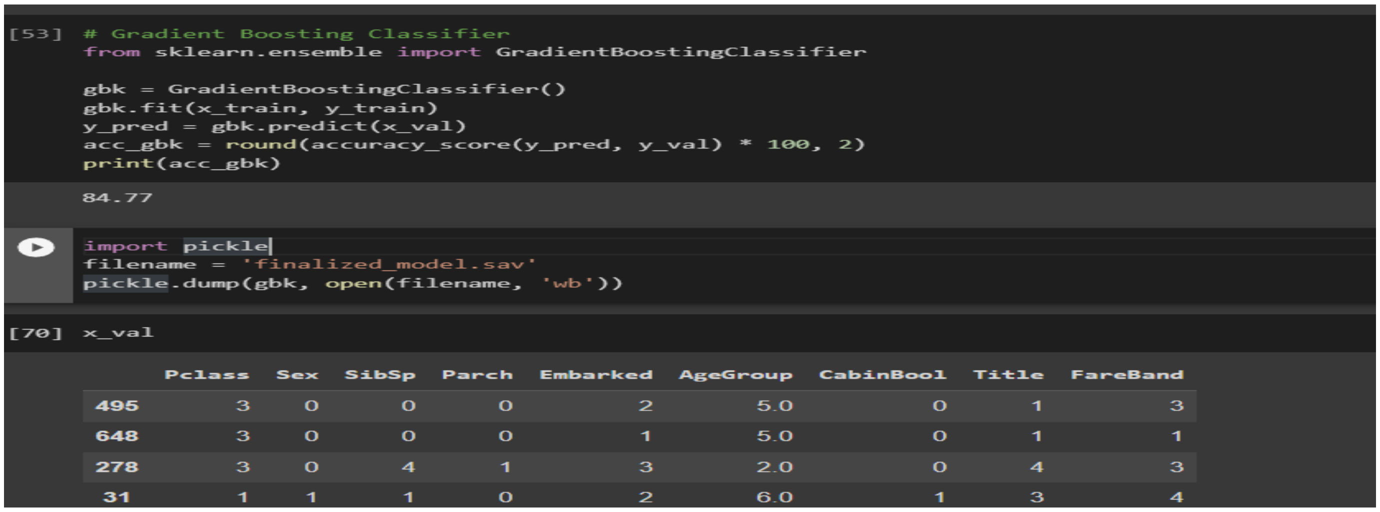
Chose passenger class: first class
Chose passenger gender: Male
Passenger Traveling with:: Alone
Number of (parents+childeren) traveling with the traveler: 1
Passenger embarked in: Southampton
Age group: Baby
The passenger had a cabin number: Yes
Title: Mr
Chose Fare Band: Cheapest

This passenger had died with chance of: 97%

The dataset and the code for EDA, Data cleaning, and model training are available on Kaggle

https://www.kaggle.com/code/nadintamer/titanic-survival-predictions-beginner/notebook

skipping the preliminary steps of the notebook,In the final step, the notebook is evaluating different models and finally picks out the gradient-boosting classifier as the best model. And so did I. I saved the trained model as a pickle file to be loaded by the server.



```
[53] # Gradient Boosting Classifier
     from sklearn.ensemble import GradientBoostingClassifier

     gbk = GradientBoostingClassifier()
     gbk.fit(x_train, y_train)
     y_pred = gbk.predict(x_val)
     acc_gbk = round(accuracy_score(y_pred, y_val) * 100, 2)
     print(acc_gbk)

     84.77
```

```
import pickle
filename = 'finalized_model.sav'
pickle.dump(gbk, open(filename, 'wb'))
```

```
[70] x_val
```

	Pclass	Sex	SibSp	Parch	Embarked	AgeGroup	CabinBool	Title	FareBand
495	3	0	0	0	2	5.0	0	1	3
648	3	0	0	0	1	5.0	0	1	1
278	3	0	4	1	3	2.0	0	4	3
31	1	1	1	0	2	6.0	1	3	4

Based on the above snapshot of the notebook it is clear that for predicting, we need 9 features, which include:

Pclass (Passenger Class)

SibSp(passenger traveling alone, with a sibling, or with a spouse)

Parch(number of parents and children with the traveler

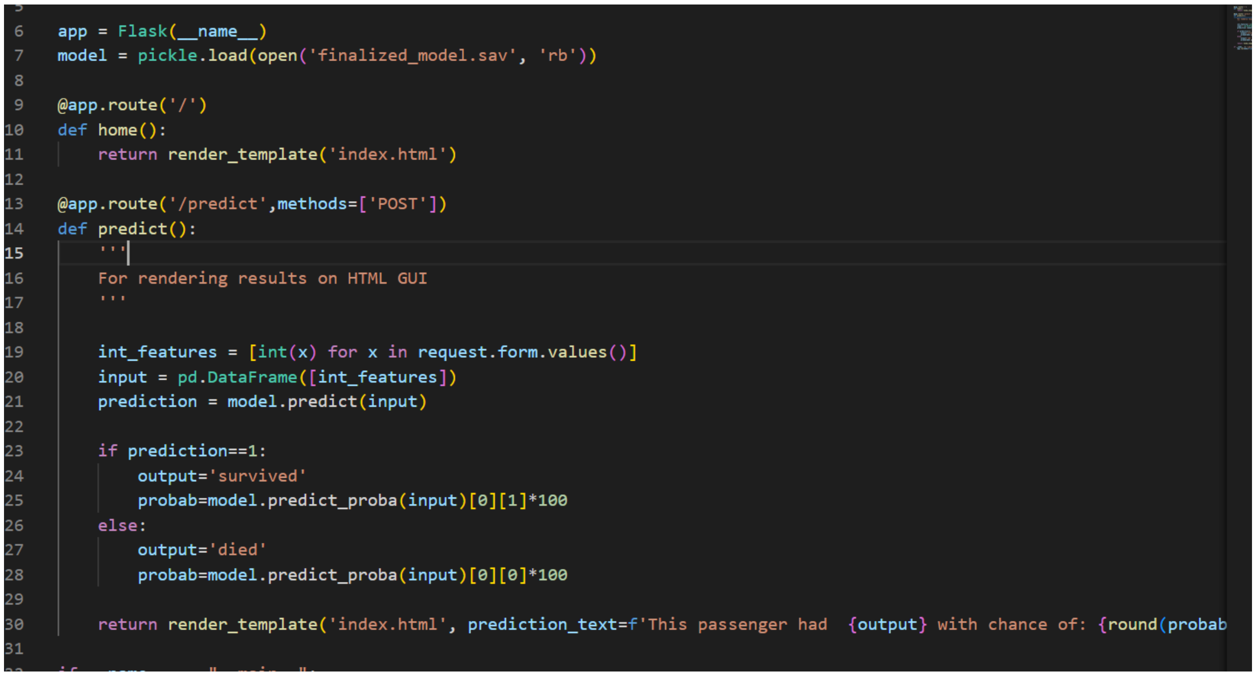
Embarked (Where did passenger embark from)

Cabinbool If the passenger had a cabin number

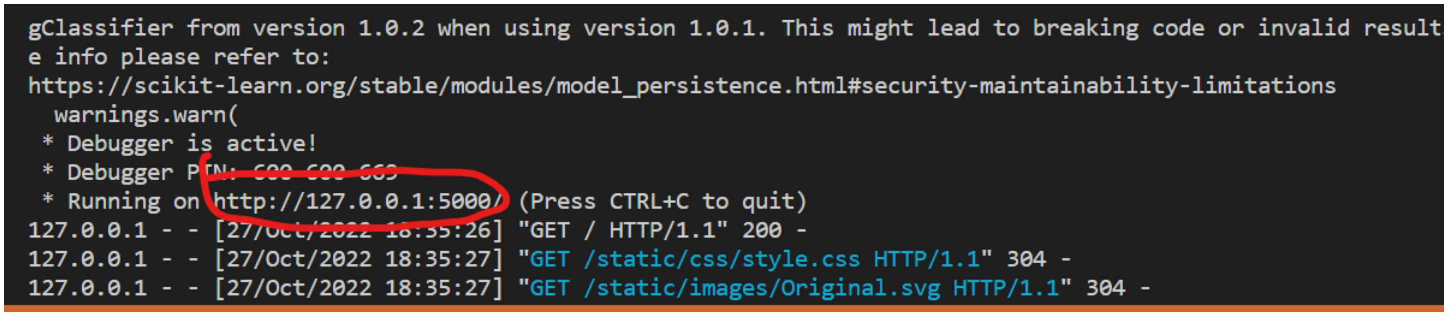
Sex, Age group, Title and Fare bond

Note: All these inputs are mapped to integers in the training code as well as descriptive texts to integers as model inputs for prediction in the web application.

After understanding our problem, data, training on different models and saving the best model,then there is deployment on the flask. Here is a snapshot of our server. After running the following code it will provide us with a link to our application



```python
app = Flask(__name__)
model = pickle.load(open('finalized_model.sav', 'rb'))

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

@app.route('/predict',methods=['POST'])
def predict():
    '''
    For rendering results on HTML GUI
    '''

    int_features = [int(x) for x in request.form.values()]
    input = pd.DataFrame([int_features])
    prediction = model.predict(input)

    if prediction==1:
        output='survived'
        probab=model.predict_proba(input)[0][1]*100
    else:
        output='died'
        probab=model.predict_proba(input)[0][0]*100

    return render_template('index.html', prediction_text=f'This passenger had  {output} with chance of: {round(probab
```

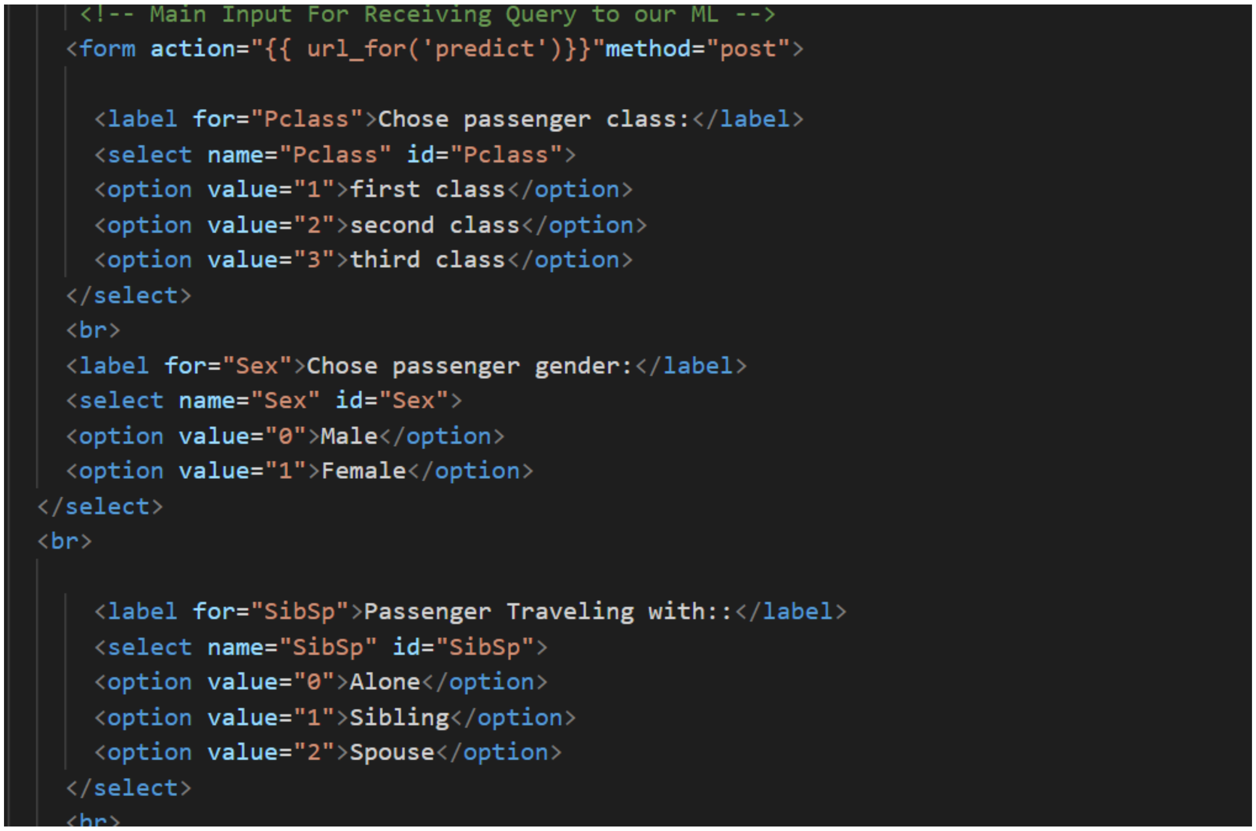


```
gClassifier from version 1.0.2 when using version 1.0.1. This might lead to breaking code or invalid result
e info please refer to:
https://scikit-learn.org/stable/modules/model_persistence.html#security-maintainability-limitations
  warnings.warn(
 * Debugger is active!
 * Debugger PIN: [illegible]
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [27/Oct/2022 18:35:26] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [27/Oct/2022 18:35:27] "GET /static/css/style.css HTTP/1.1" 304 -
127.0.0.1 - - [27/Oct/2022 18:35:27] "GET /static/images/Original.svg HTTP/1.1" 304 -
```

In the server, The predict function takes the int features from the HTML form as a list and feeds this list as input to our loaded model for prediction and returning the value (survived or died) as well as chance of survival or death.

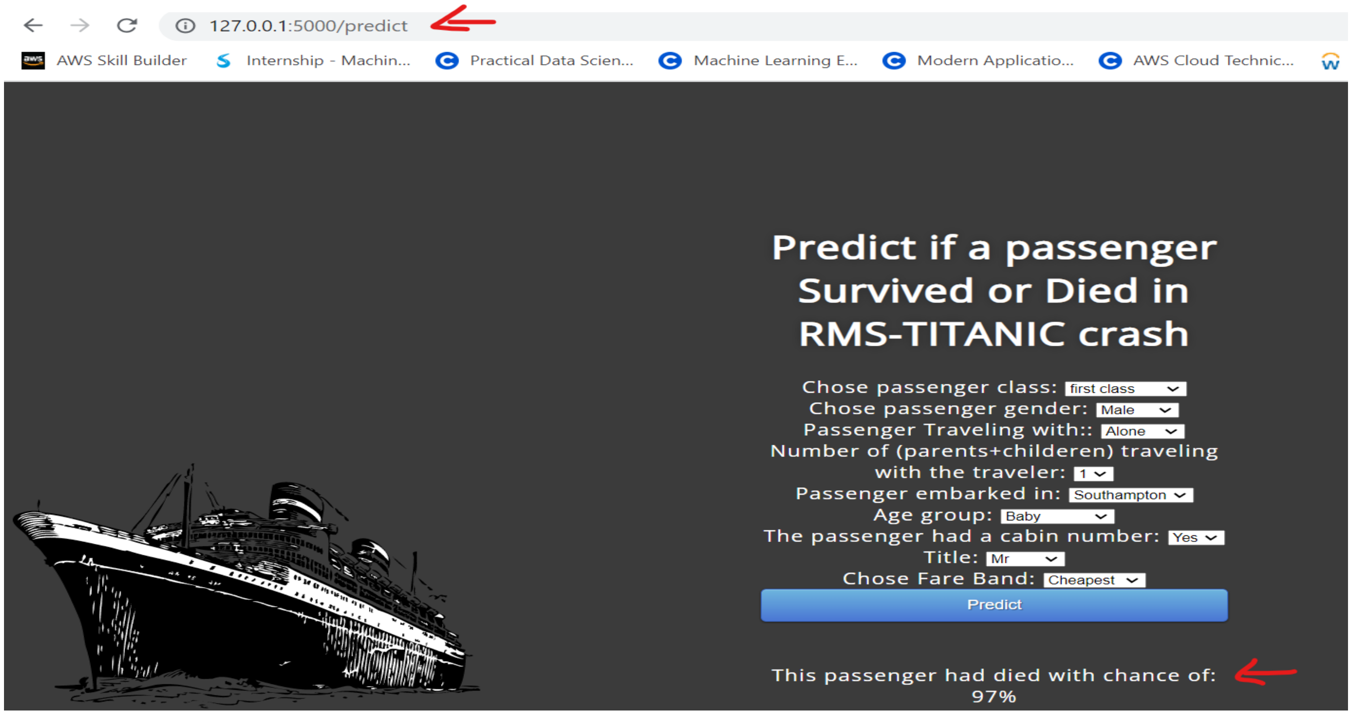
HTML:

The input format of the HTML form is select, where the user selects a descriptive value mapped to integers which are understandable values for our model as following snapshot:



```
  <!-- Main Input For Receiving Query to our ML -->
  <form action="{{ url_for('predict')}}"method="post">

    <label for="Pclass">Chose passenger class:</label>
    <select name="Pclass" id="Pclass">
    <option value="1">first class</option>
    <option value="2">second class</option>
    <option value="3">third class</option>
  </select>
  <br>
  <label for="Sex">Chose passenger gender:</label>
  <select name="Sex" id="Sex">
  <option value="0">Male</option>
  <option value="1">Female</option>
</select>
<br>

    <label for="SibSp">Passenger Traveling with::</label>
    <select name="SibSp" id="SibSp">
    <option value="0">Alone</option>
    <option value="1">Sibling</option>
    <option value="2">Spouse</option>
  </select>
```

In the end, the web application look like the following :



127.0.0.1:5000/predict

Predict if a passenger Survived or Died in RMS-TITANIC crash

Chose passenger class: first class
Chose passenger gender: Male
Passenger Traveling with:: Alone
Number of (parents+childeren) traveling with the traveler: 1
Passenger embarked in: Southampton
Age group: Baby
The passenger had a cabin number: Yes
Title: Mr
Chose Fare Band: Cheapest

This passenger had died with chance of: 97%