

DATA GLACIER SIMPLE FLASK APPLICATION (DEPLOYMENT ON FLASK):

Name: Blaise Papa

Batch Code: LISUM01

Submission date: 3rd July 2021

GET TOY DATA AND LOAD

The model was developed around loan prediction data.

The data is collected on customer who apply for loans, the model is based to classify the customers between two classes; those whose loans are accepted and those whose loans are rejected.

Loan_ID																	
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Loan_ID	Gender	Married	Dependent	Education	Self_Emp	Applicant	Coapplicant	Loan_Amo	Loan_Amo	Credit_Hist	Property	_fLoan_Status				
2	LP001002	Male	No	0	Graduate	No	5849	0	360	1	Urban	Y					
3	LP001003	Male	Yes	1	Graduate	No	4583	1508	128	360	1	Rural	N				
4	LP001005	Male	Yes	0	Graduate	Yes	3000	0	66	360	1	Urban	Y				
5	LP001006	Male	Yes	0	Not Gradua	No	2583	2358	120	360	1	Urban	Y				
6	LP001008	Male	No	0	Graduate	No	6000	0	141	360	1	Urban	Y				
7	LP001011	Male	Yes	2	Graduate	Yes	5417	4196	267	360	1	Urban	Y				
8	LP001013	Male	Yes	0	Not Gradua	No	2333	1516	95	360	1	Urban	Y				
9	LP001014	Male	Yes	3+	Graduate	No	3036	2504	158	360	0	Semiurban	N				
10	LP001018	Male	Yes	2	Graduate	No	4006	1526	168	360	1	Urban	Y				
11	LP001020	Male	Yes	1	Graduate	No	12841	10968	349	360	1	Semiurban	N				
12	LP001024	Male	Yes	2	Graduate	No	3200	700	70	360	1	Urban	Y				
13	LP001027	Male	Yes	2	Graduate	No	2500	1840	109	360	1	Urban	Y				
14	LP001028	Male	Yes	2	Graduate	No	3073	8106	200	360	1	Urban	Y				
15	LP001029	Male	No	0	Graduate	No	1853	2840	114	360	1	Rural	N				
16	LP001030	Male	Yes	2	Graduate	No	1299	1086	17	120	1	Urban	Y				
17	LP001032	Male	No	0	Graduate	No	4950	0	125	360	1	Urban	Y				
18	LP001034	Male	No	1	Not Gradua	No	3596	0	100	240	1	Urban	Y				
19	LP001036	Female	No	0	Graduate	No	2510	0	76	360	0	Urban	N				

MODEL BUILDING

The model is built and saved in the model.py file

Loading data to python and converting into a data frame we can manipulate.

```
1  """
2
3  | simple random forest regressor model to predict loan eligibility
4
5
6  """
7  | You, 6 minutes ago • Simple flask application on loan prediction data
8  import pandas as pd
9  import numpy as np
10 import pickle
11
12
13 train=pd.read_csv('C:\Users\blais\Desktop\Data Glacier\Task2\DataGlacier\week4\train_ctrUa4K.csv')
14
15
16 def dropNullCols(data):
17     print("Deleting in progress")
18     data.dropna(inplace=True)
19
20     return data
21
22 dropNullCols(train)
23 # dropNullCols(test)
24
```

Simple data preprocessing steps to prepare data for model training. In this case we simply dropped all null column along with the 'loan_satus' column and label encoded the object columns.

```
model.py 6
model.py > ...
25
26 from sklearn.preprocessing import LabelEncoder
27 from sklearn.preprocessing import OrdinalEncoder
28
29 le=LabelEncoder()
30 oe=OrdinalEncoder()
31
32
33 train
34 train['Loan_Status']=train['Loan_Status'].replace({'Y':1,'N':0})
35 train['Gender']=le.fit_transform(train['Gender'])
36 train['Education']=le.fit_transform(train['Education'])
37 train['Dependents']=le.fit_transform(train['Dependents'])
38 train['Self_Employed']=le.fit_transform(train['Self_Employed'])
39 train['Property_Area']=le.fit_transform(train['Property_Area'])
40 train['Married']=le.fit_transform(train['Married'])
41 train
42
43
44 train.drop(labels=['Loan_ID'],inplace=True,axis=1)
45
46
47 X=train.drop(labels=['Loan_Status'],axis=1)
48 y=train['Loan_Status'].values
49
```

MODEL TRAINING

For this binary classification model, we shall employ the random forest classifier, a well-known ensemble model. We also incorporate accuracy, f1 score and roc score as model evaluation metrics

```
model.py 6
model.py > ...
41 train
42
43
44 train.drop(labels=['Loan_ID'],inplace=True,axis=1)
45
46
47 X=train.drop(labels=['Loan_Status'],axis=1)
48 y=train['Loan_Status'].values
49
50
51 from sklearn.ensemble import RandomForestClassifier
52 from sklearn.model_selection import train_test_split
53 from sklearn.metrics import accuracy_score,confusion_matrix,f1_score,roc_auc_score
54
55
56 X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=.3,random_state=0)
57
58
59 rf=RandomForestClassifier()
60
61 rf.fit(X_train,y_train)
62
63 print(rf.score(X_train,y_train))
64 print(rf.score(X_test,y_test))
65
66
67 predictions=rf.predict(X_test)
68
```

MODEL SERIALIZATION.

Once satisfied with our model we serialize it into a pickle file which will enable us to deploy the trained model.

```
predictions=rf.predict(X_test)

print(accuracy_score(y_test,predictions))

pickle.dump(rf,open('model.pkl','wb'))
```

BUILD SIMPLE HTML WEBSITE

We create a simple website that users are able to interact with.

```
<body>
  <div id="login-form-container">
    <form action="{ url_for('predict')}" method="post">
      <div class="card" style="width: 400px">
        <div class="card-content">
          <div class="media">
            <div class="is-size-4 has-text-centered">Loan Prediction Model</div>
          </div>
          <div class="content">

            <div class="field">
              <p class="control">
                Gender: <select id="gender" name="gender">
                  <option value=1>Male</option>
                  <option value=0>Female</option>
                </select>
              </p>
            </div>

            <div class="field">
              <p class="control">
                Married:
                <select id="married" name="married">
                  <option value=1>Married</option>
                  <option value=0>Not Married</option>
                </select>
              </p>
            </div>

          </div>
        </div>
      </div>
    </form>
  </div>
</body>
```

BUILD FLASK APPLICATION

We create a simple flask application that will serve and deploy our model.

Model deserialization

We define our flask application and then deserialize the model we earlier trained.

We create a default router which will be rendered when the default endpoint is called in this case the default endpoint('/') renders the homepage.html file which is a simple website we had earlier created.

```
model.py 6 • app.py 1 X
app.py > ...
You, 9 minutes ago | 1 author (You)
1 from flask import Flask,request,render_template,jsonify
2 import numpy as np
3 import pickle
4
5
6 app=Flask(__name__)
7 model=pickle.load(open('model.pkl','rb'))
8
9
10
11
12
13 @app.route('/')
14
15 def homepage():
16     return render_template('homepage.html')
17
18
```

BUILT ROUTES AND FUNCTIONS

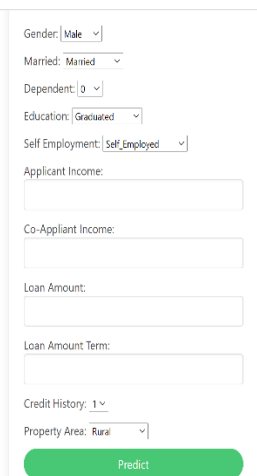
We define other routes that enable us to call various endpoints in our application

```
model.py 6 • app.py 1 X
app.py > ...
28
29     output=round(prediction[0],2)
30
31     return render_template('homepage.html',predictions='The loan prediction is {}'.format(output))
32
33
34 @app.route('/results',methods=['POST'])
35 def results():
36
37     data = request.get_json(force=True)
38     prediction = model.predict([np.array(list(data.values()))])
39
40     output = prediction[0]
41     return jsonify(output)
42
43
44
45 if __name__ == "__main__":
46     app.run(debug=True)
```

```
model.py 6 • request.py 1 x
request.py > ...
You, 12 minutes ago | 1 author (You)
1 import requests You, 12 minutes ago • Simple flask application on loan prediction data
2
3 url = 'http://localhost:8080/results'
4 r = requests.post(url,json={'rate':5, 'sales_in_first_month':200, 'sales_in_second_month':400})
5
6 print(r.json())
```

DEPLOYED APPLICATION:

This is the rendered application.



The form contains the following fields:

- Gender: Male (dropdown)
- Married: Married (dropdown)
- Dependent: 0 (dropdown)
- Education: Graduated (dropdown)
- Self Employment: Self_Employed (dropdown)
- Applicant Income: (text input)
- Co-Applicant Income: (text input)
- Loan Amount: (text input)
- Loan Amount Term: (text input)
- Credit History: 1 (dropdown)
- Property Area: Rural (dropdown)
- Predict (green button)

The use is expected to fill in the form with details of the customer.

Gender:

Married:

Dependent:

Education:

Self Employment:

Applicant Income:

Co-Applicant Income:

Loan Amount:

Loan Amount Term:

Credit History:

Property Area:

The model receives input and produces a prediction.

The loan prediction is 1