DATA GLACIER SIMPLE APPLICATION (DEPLOYMENT ON CLOUD):

Name: Blaise Papa

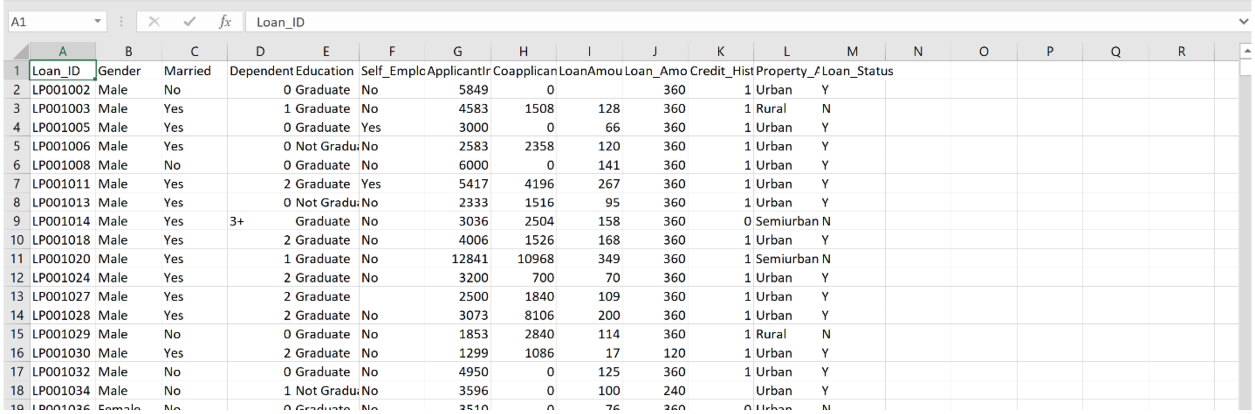
Batch Code: LISUM01

Submission date:11th 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.



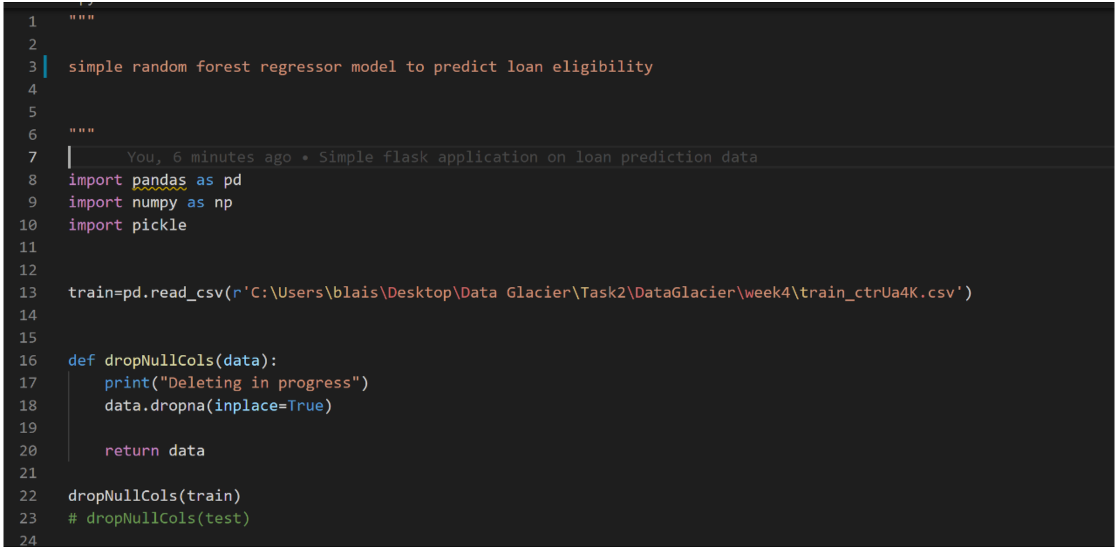
A1 Loan_ID

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Loan_ID	Gender	Married	Dependent	Education	Self_Emplo	ApplicantIr	Coapplican	LoanAmou	Loan_Amo	Credit_Hist	Property_A	Loan_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 Gradu	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 Gradu	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		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 Gradu	No	3596	0	100	240		Urban	Y
19	LP001036	Female	No	0	Graduate	No	3510	0	76	360	0	Urban	N

MODEL BUILDING

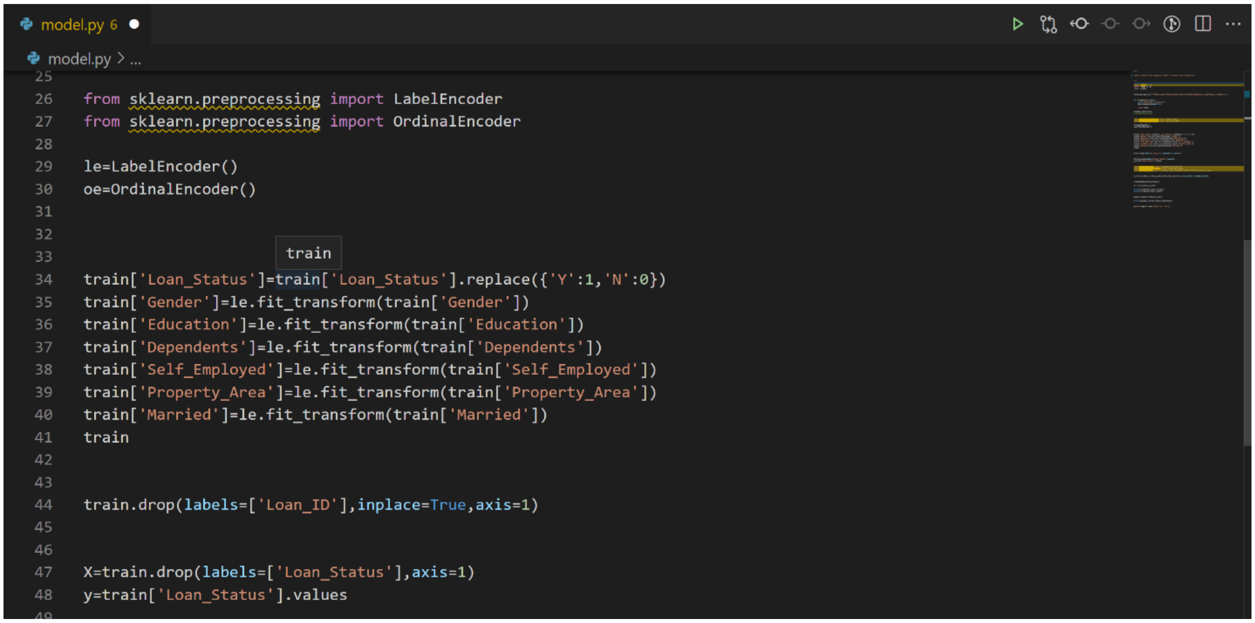
The model is built and saved in the model.py file

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



```
"""

simple random forest regressor model to predict loan eligibility


"""
          You, 6 minutes ago • Simple flask application on loan prediction data
import pandas as pd
import numpy as np
import pickle


train=pd.read_csv(r'C:\Users\blais\Desktop\Data Glacier\Task2\DataGlacier\week4\train_ctrUa4K.csv')


def dropNullCols(data):
    print("Deleting in progress")
    data.dropna(inplace=True)

    return data

dropNullCols(train)
# dropNullCols(test)

```

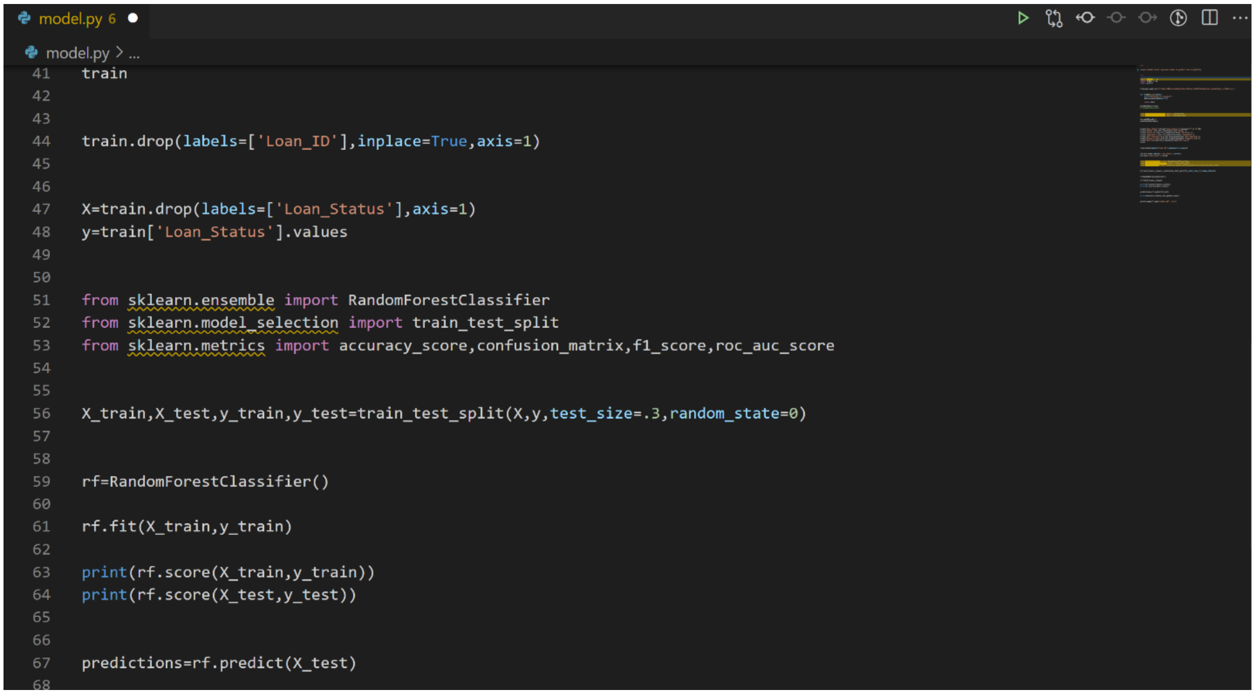
Simple data preprocessing steps to prepare data for model training. In this case we simply dropped all null column along with the ‘loan_satus’ column and label encoded the object columns.



```python

from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OrdinalEncoder

le=LabelEncoder()
oe=OrdinalEncoder()



train['Loan_Status']=train['Loan_Status'].replace({'Y':1,'N':0})
train['Gender']=le.fit_transform(train['Gender'])
train['Education']=le.fit_transform(train['Education'])
train['Dependents']=le.fit_transform(train['Dependents'])
train['Self_Employed']=le.fit_transform(train['Self_Employed'])
train['Property_Area']=le.fit_transform(train['Property_Area'])
train['Married']=le.fit_transform(train['Married'])
train


train.drop(labels=['Loan_ID'],inplace=True,axis=1)


X=train.drop(labels=['Loan_Status'],axis=1)
y=train['Loan_Status'].values
```

MODEL TRAINING

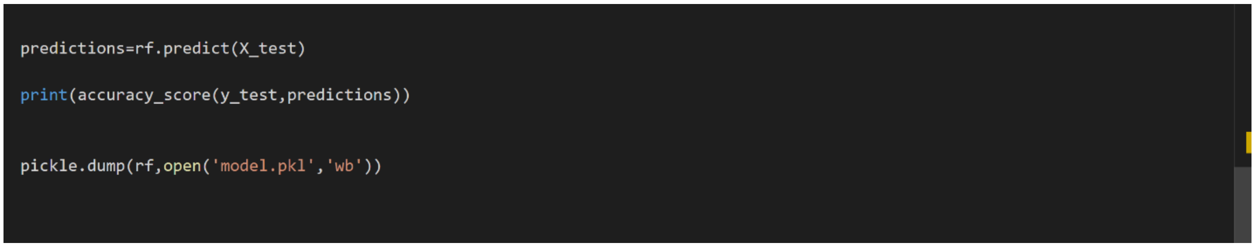
For this binary classification model, we shall employ the random forest classifier, a well-known ensemble model. We also incorporate accuracy, f1 score and roc score as model evaluation metrics



```python
train


train.drop(labels=['Loan_ID'],inplace=True,axis=1)


X=train.drop(labels=['Loan_Status'],axis=1)
y=train['Loan_Status'].values


from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,confusion_matrix,f1_score,roc_auc_score


X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=.3,random_state=0)


rf=RandomForestClassifier()

rf.fit(X_train,y_train)

print(rf.score(X_train,y_train))
print(rf.score(X_test,y_test))


predictions=rf.predict(X_test)
```

MODEL SERIALIZATION.

Once satisfied with out 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'))
```

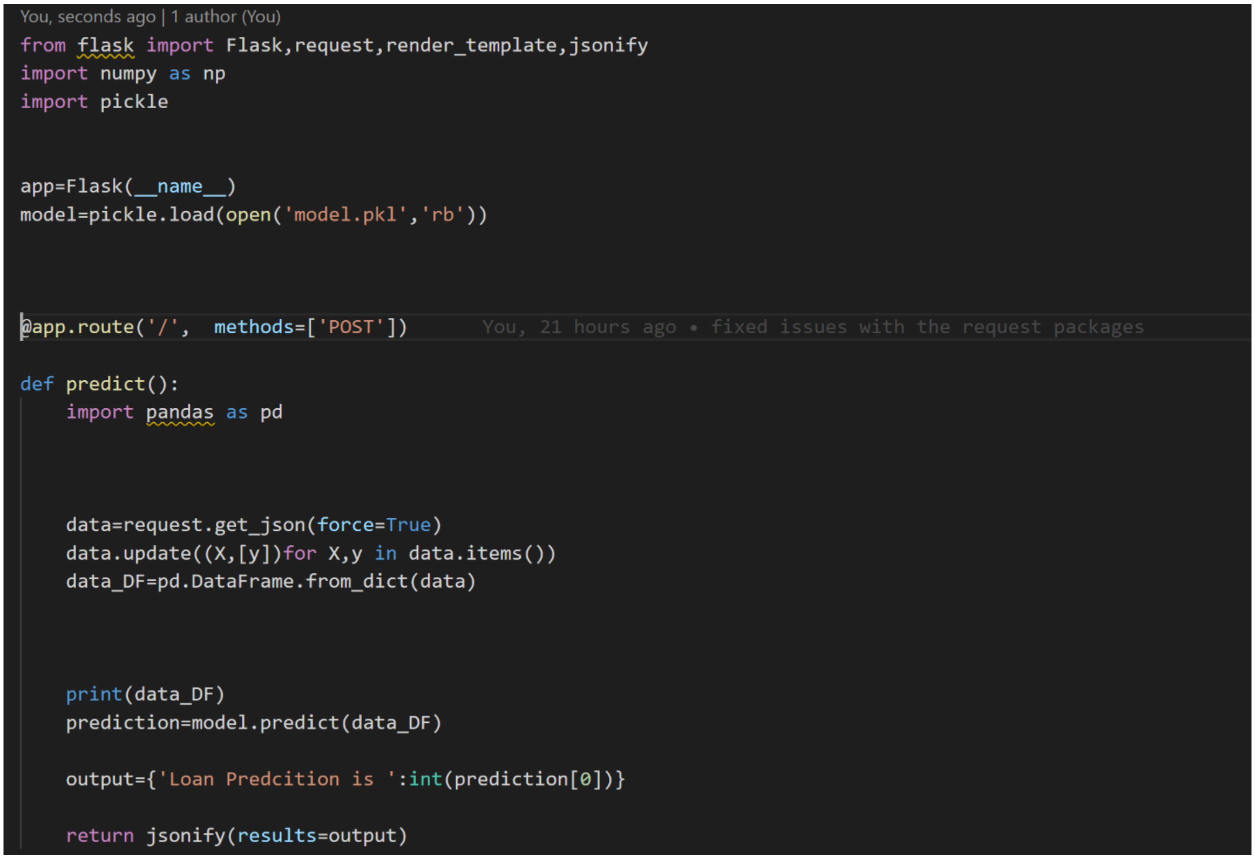
BUILD FLASK APPLICATION

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.

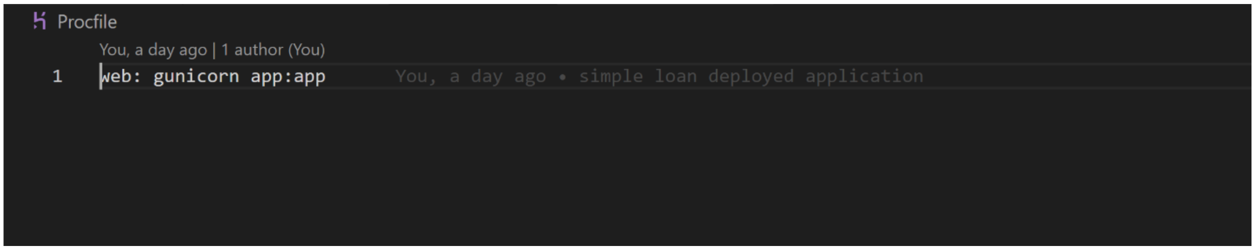
We create a default router which will be rendered when the default endpoint is called in this case the default endpoint(‘/’) receives data in json format and converts it to a data frame which can be processed by the model.



```
You, seconds ago | 1 author (You)
from flask import Flask,request,render_template,jsonify
import numpy as np
import pickle


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



@app.route('/',  methods=['POST'])      You, 21 hours ago • fixed issues with the request packages

def predict():
    import pandas as pd



    data=request.get_json(force=True)
    data.update((X,[y])for X,y in data.items())
    data_DF=pd.DataFrame.from_dict(data)



    print(data_DF)
    prediction=model.predict(data_DF)

    output={'Loan Predcition is ':int(prediction[0])}

    return jsonify(results=output)
```

CREATE A PROCFILE

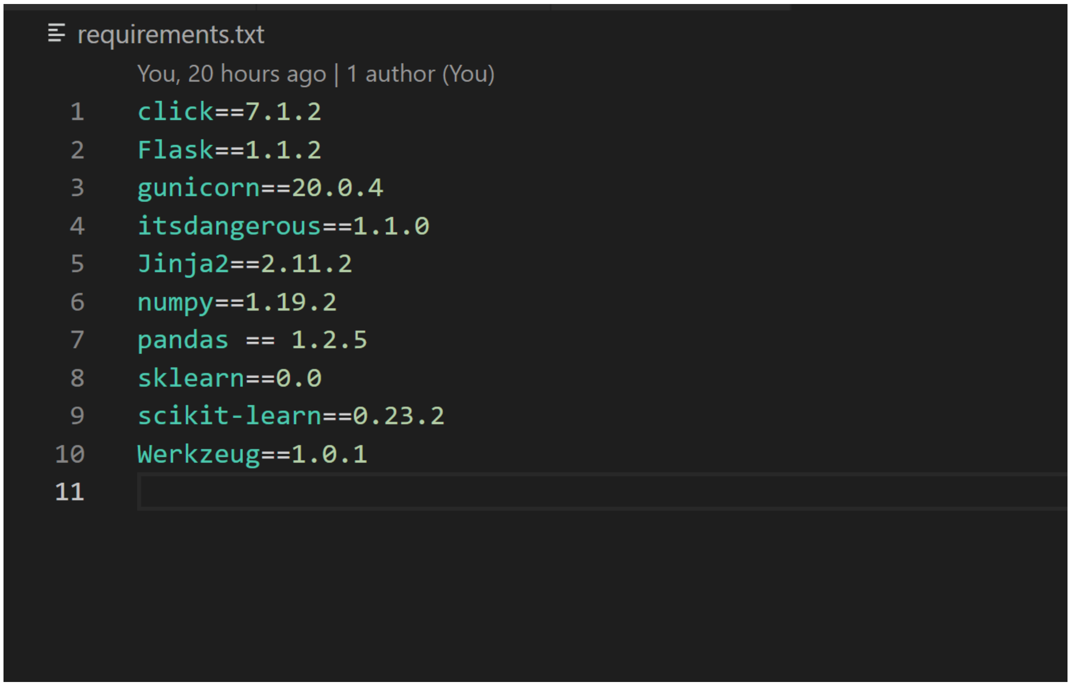
This specifies the commands that are executed by Heroku app on startup in this case it should run the app.py script that we have created.



```
web: gunicorn app:app        You, a day ago • simple loan deployed application
```

CREATE A REQUIREMENTS.TXT FILE

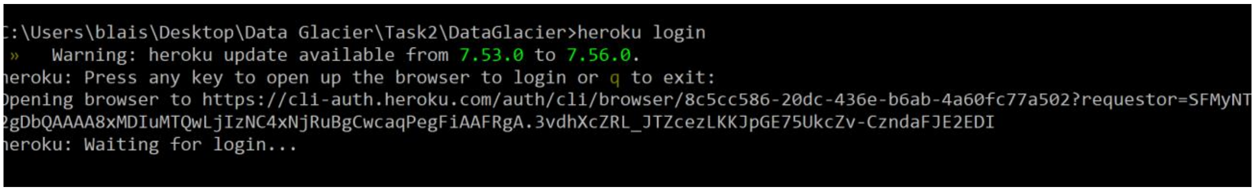
This file should contain all the dependencies/ libraries that are necessary to succefully deploy the application.



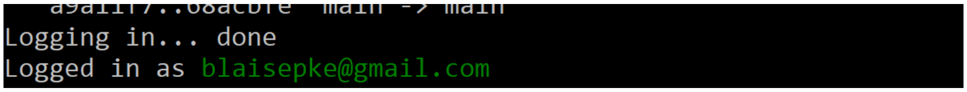
```
click==7.1.2
Flask==1.1.2
gunicorn==20.0.4
itsdangerous==1.1.0
Jinja2==2.11.2
numpy==1.19.2
pandas == 1.2.5
sklearn==0.0
scikit-learn==0.23.2
Werkzeug==1.0.1
```

DEPLOYING TO HEROKU

Before deployment we require a Heroku account, since we already have one we simply login to it.



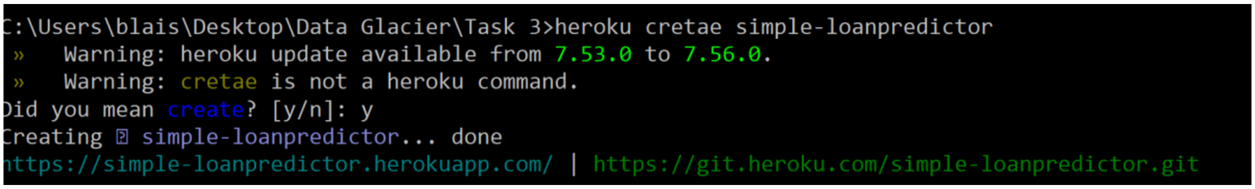
```
:\Users\blais\Desktop\Data Glacier\Task2\DataGlacier>heroku login
 »   Warning: heroku update available from 7.53.0 to 7.56.0.
eroku: Press any key to open up the browser to login or q to exit:
pening browser to https://cli-auth.heroku.com/auth/cli/browser/8c5cc586-20dc-436e-b6ab-4a60fc77a502?requestor=SFMyNT
gDbQAAAA8xMDIuMTQwLjIzNC4xNjRuBgCwcaqPegFiAAFRgA.3vdhXcZRL_JTZcezLKKJpGE75UkcZv-CzndaFJE2EDI
eroku: Waiting for login...
```



```
   a9a11f7..68acbfe  main -> main
Logging in... done
Logged in as blaisepke@gmail.com
```

CREATE AN APPLICATION.

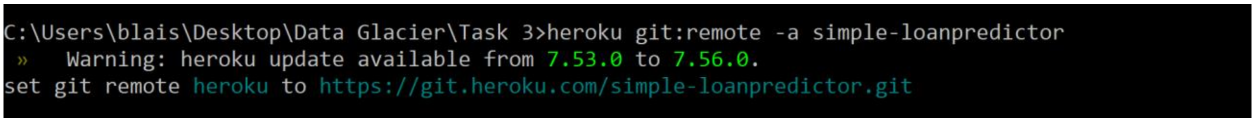
We create a new application which we will deploy to the cloud



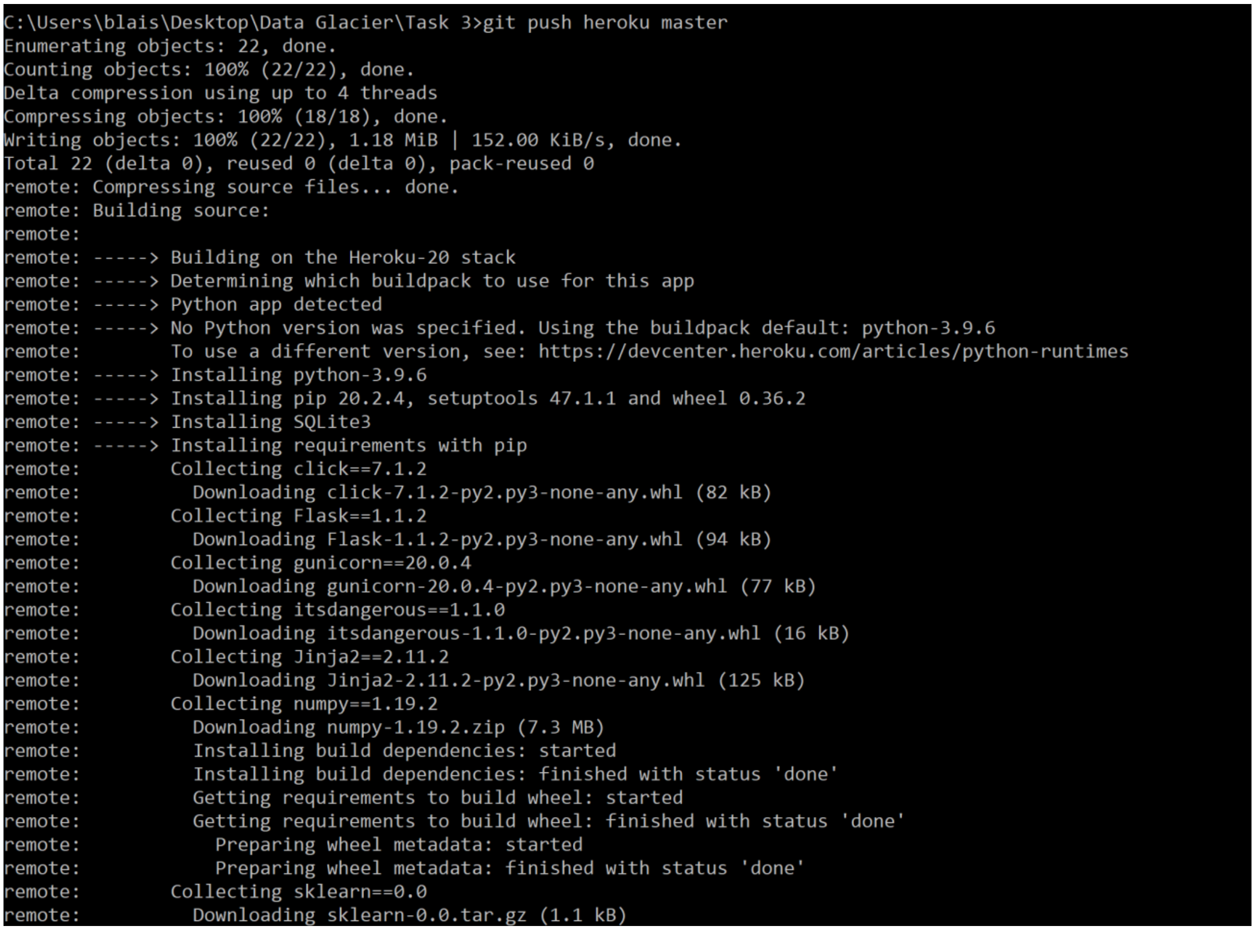
```
C:\Users\blais\Desktop\Data Glacier\Task 3>heroku cretae simple-loanpredictor
 »   Warning: heroku update available from 7.53.0 to 7.56.0.
 »   Warning: cretae is not a heroku command.
Did you mean create? [y/n]: y
Creating ⍰ simple-loanpredictor... done
https://simple-loanpredictor.herokuapp.com/ | https://git.heroku.com/simple-loanpredictor.git
```

DEPLOY MODEL ON HEROKU

To deploy the application, we first create a remote repo with the app name so we will be able to push to



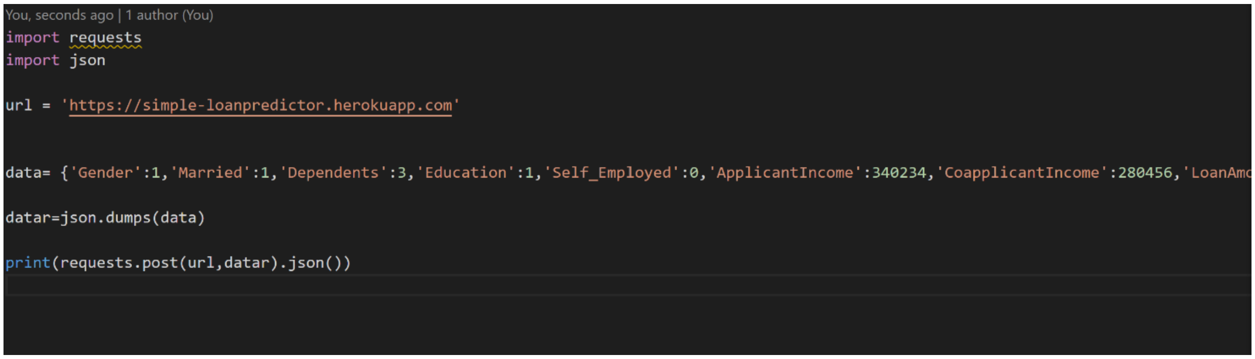
```
C:\Users\blais\Desktop\Data Glacier\Task 3>heroku git:remote -a simple-loanpredictor
 »   Warning: heroku update available from 7.53.0 to 7.56.0.
set git remote heroku to https://git.heroku.com/simple-loanpredictor.git
```



```
C:\Users\blais\Desktop\Data Glacier\Task 3>git push heroku master
Enumerating objects: 22, done.
Counting objects: 100% (22/22), done.
Delta compression using up to 4 threads
Compressing objects: 100% (18/18), done.
Writing objects: 100% (22/22), 1.18 MiB | 152.00 KiB/s, done.
Total 22 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Building on the Heroku-20 stack
remote: -----> Determining which buildpack to use for this app
remote: -----> Python app detected
remote: -----> No Python version was specified. Using the buildpack default: python-3.9.6
remote:        To use a different version, see: https://devcenter.heroku.com/articles/python-runtimes
remote: -----> Installing python-3.9.6
remote: -----> Installing pip 20.2.4, setuptools 47.1.1 and wheel 0.36.2
remote: -----> Installing SQLite3
remote: -----> Installing requirements with pip
remote:        Collecting click==7.1.2
remote:          Downloading click-7.1.2-py2.py3-none-any.whl (82 kB)
remote:        Collecting Flask==1.1.2
remote:          Downloading Flask-1.1.2-py2.py3-none-any.whl (94 kB)
remote:        Collecting gunicorn==20.0.4
remote:          Downloading gunicorn-20.0.4-py2.py3-none-any.whl (77 kB)
remote:        Collecting itsdangerous==1.1.0
remote:          Downloading itsdangerous-1.1.0-py2.py3-none-any.whl (16 kB)
remote:        Collecting Jinja2==2.11.2
remote:          Downloading Jinja2-2.11.2-py2.py3-none-any.whl (125 kB)
remote:        Collecting numpy==1.19.2
remote:          Downloading numpy-1.19.2.zip (7.3 MB)
remote:          Installing build dependencies: started
remote:          Installing build dependencies: finished with status 'done'
remote:          Getting requirements to build wheel: started
remote:          Getting requirements to build wheel: finished with status 'done'
remote:            Preparing wheel metadata: started
remote:            Preparing wheel metadata: finished with status 'done'
remote:        Collecting sklearn==0.0
remote:          Downloading sklearn-0.0.tar.gz (1.1 kB)
```

CREATE A DUMMY API REQUEST

We create a dummy request to interact with the application to test whether it works

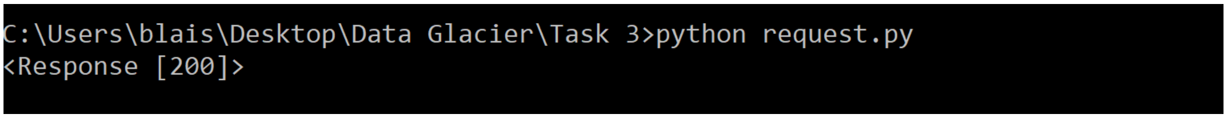


You, seconds ago | 1 author (You)

```python
import requests
import json

url = 'https://simple-loanpredictor.herokuapp.com'


data= {'Gender':1,'Married':1,'Dependents':3,'Education':1,'Self_Employed':0,'ApplicantIncome':340234,'CoapplicantIncome':280456,'LoanAmo

datar=json.dumps(data)

print(requests.post(url,datar).json())
```

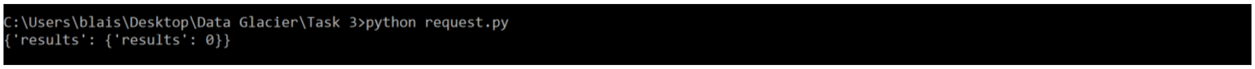
CHECK FOR RESPONSE

We view the server response



```
C:\Users\blais\Desktop\Data Glacier\Task 3>python request.py
<Response [200]>
```

The server response looks good so we check for the predictions



```
C:\Users\blais\Desktop\Data Glacier\Task 3>python request.py
{'results': {'results': 0}}
```