DATA GLACIER

WEEK 4: DEPLOYMENT ON FLASK

IVAN CAMILO SABOGAL MORENO
LISUM13

27/09/2022

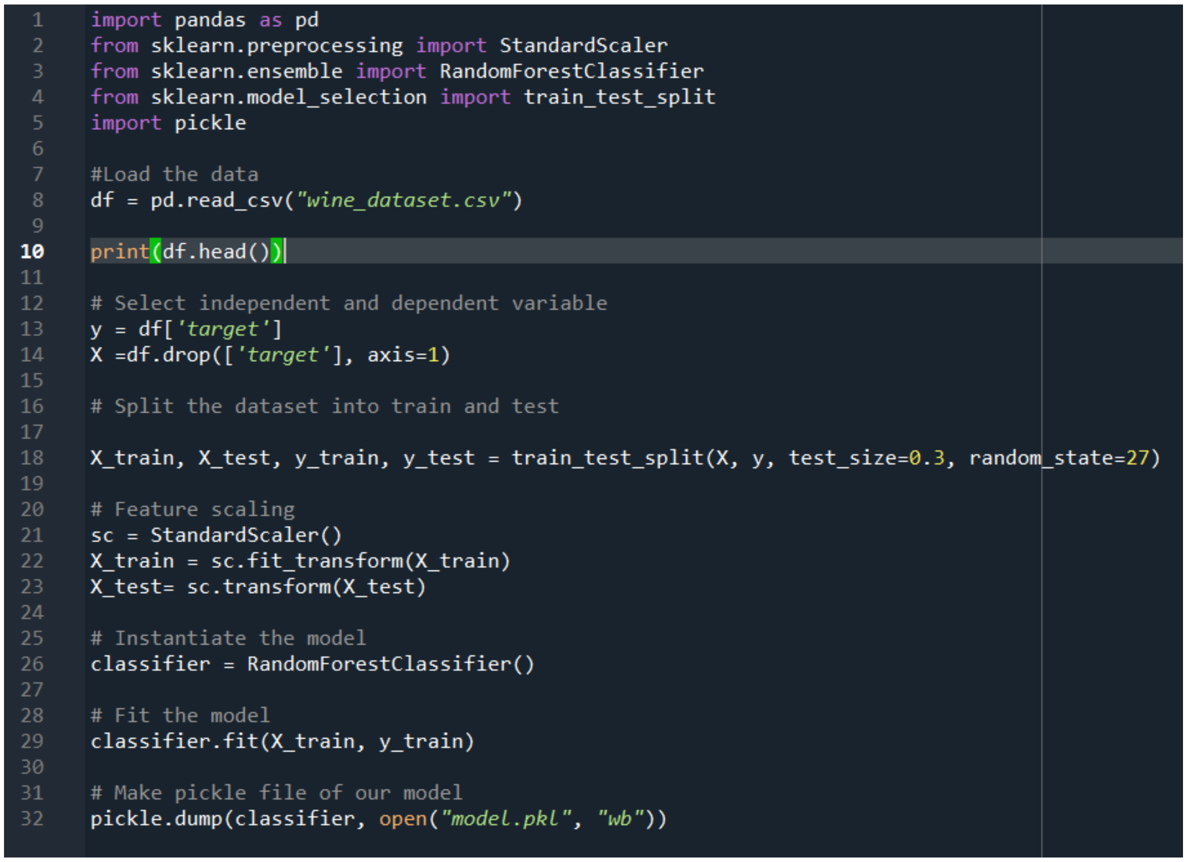
SUBMITTED: DATA GLACIER

DATASET

For this work we used the wine recognition dataset, this is the results of a chemical análisis of wine grown in the same región in Italy and there're 3 different class of wine and there're 13 feature that help to differentiate wines.

- **Features:**
  - Alcohol
  -Malic acid
  -Ash
  -Alcalinity of ash
  -Magnesium
  -Total phenols
  -Flavanoids
  -Nonflavanoid phenols
  -Proanthocyanins
  -Color intensity
  -Hue
  -OD280/OD315
  -Proline

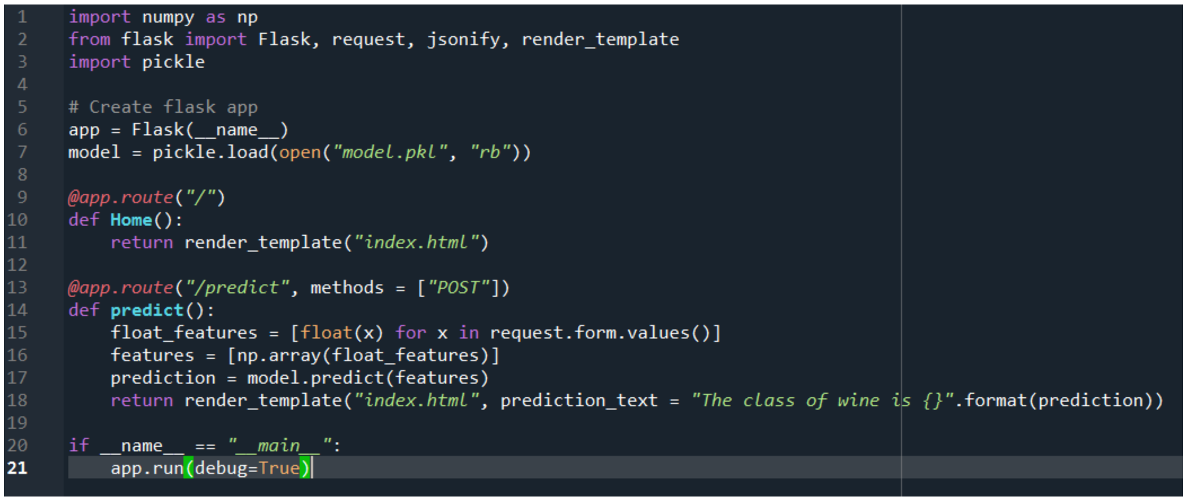
BUILD MODEL AND SAVE MODEL



```python
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import pickle

#Load the data
df = pd.read_csv("wine_dataset.csv")

print(df.head())

# Select independent and dependent variable
y = df['target']
X =df.drop(['target'], axis=1)

# Split the dataset into train and test

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=27)

# Feature scaling
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test= sc.transform(X_test)

# Instantiate the model
classifier = RandomForestClassifier()

# Fit the model
classifier.fit(X_train, y_train)

# Make pickle file of our model
pickle.dump(classifier, open("model.pkl", "wb"))
```

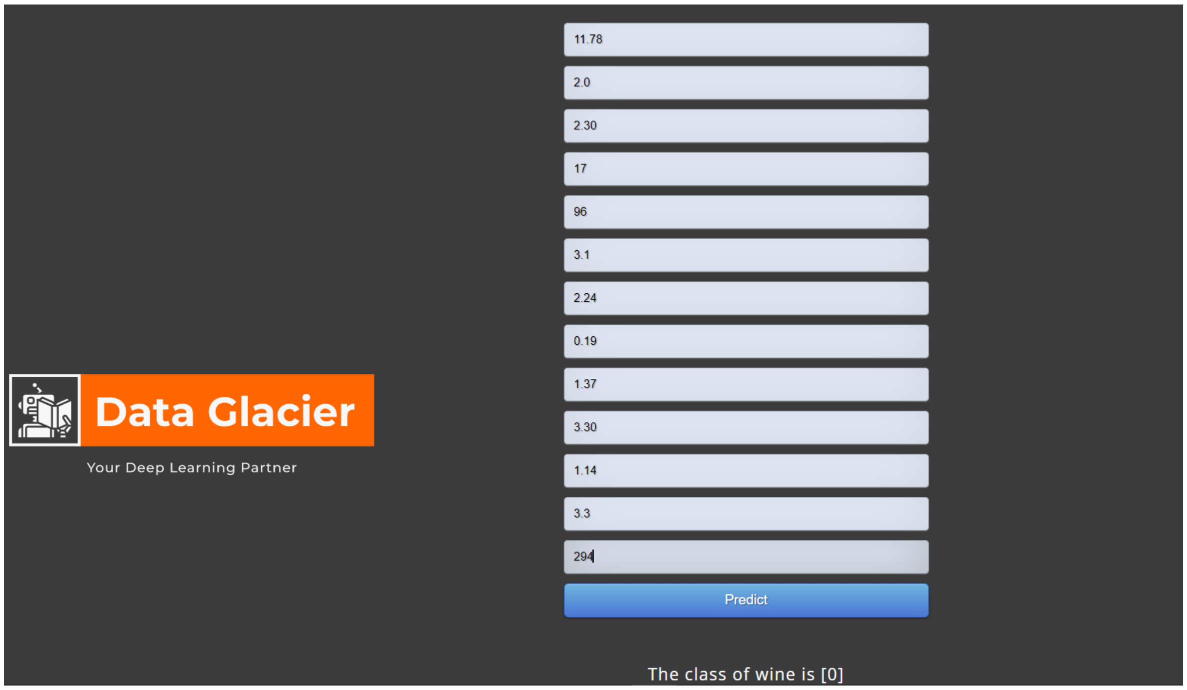
We load the libraries needed to build and save the model. Then we load the wine data, separate the target characteristics. We split our data in training and test into 70% training and 30%. We scale the features so that they do not affect the model and use randomforest to do the classification, train and then save the model.

APP.PY



```python
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle

# Create flask app
app = Flask(__name__)
model = pickle.load(open("model.pkl", "rb"))

@app.route("/")
def Home():
    return render_template("index.html")

@app.route("/predict", methods = ["POST"])
def predict():
    float_features = [float(x) for x in request.form.values()]
    features = [np.array(float_features)]
    prediction = model.predict(features)
    return render_template("index.html", prediction_text = "The class of wine is {}".format(prediction))

if __name__ == "__main__":
    app.run(debug=True)
```

We load the libraries in order to deploy the Flask web application. Then we load the saved model and create the flask app. After that we take the htlm document to load the page and take the data, predict and return the result.



11.78
2.0
2.30
17
96
3.1
2.24
0.19
1.37
3.30
1.14
3.3
294

The class of wine is [0]

This is an example