Amr Osama Data Glacier LISP01 20-3-2021

1. Model building and saving

creating the simple machine learning model model.py, A simple logistic regression which classifies whether the client is going to purchase the product after seeing the add or not.

```
File Edit Search Source Run Debug Consoles Projects Tools View Help
                         D:\My Projects\Data Glacier\Week 4\LR model\simple_LR.py
E
        # In[1]:
        import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
        import pickle
        dataset = pd.read_csv('D:\\My Projects\\Data Glacier\\Week 4\\LR model\\Social_Network_Ads.csv')
        X = dataset.iloc[:, :-1].values
        y = dataset.iloc[:, -1].values
        from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1, random_state = 0)
        # ## Feature Scaling
        from sklearn.preprocessing import StandardScaler
        sc = StandardScaler()
        X_train = sc.fit_transform(X_train)
        X_test = sc.transform(X_test)
        # ## Training the Logistic Regression model on the Training set
        from sklearn.linear_model import LogisticRegression
        classifier = LogisticRegression(random_state = 0)
        classifier.fit(X_train, y_train)
        # In[8]:
        # Saving model to disk
        pickle.dump(classifier, open('model.pkl','wb'))
        # In[10]:
        model = pickle.load(open('model.pkl','rb'))
        print(model.predict([[19,1900]]))
```

2. App.py

The next part was to make an API which receives input details through GUI and predicts the output.

```
Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
                            □ □ □ C | □ □ C:\Users\10
D:\My Projects\Data Glacier\Week 4\LR model\app.py
simple_LR.py* × module.py × index.html ×
         # coding: utf-8
         # In[9]:
         import numpy as np
         from flask import Flask, request, jsonify, render_template
         import pickle
         app = Flask(__name__)
         model = pickle.load(open('model.pkl', 'rb'))
         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()]
             final_features = [np.array(int_features)]
prediction = model.predict(final_features)
             output = prediction
             return render_template('index.html', prediction_text='Purchased {}'.format(output))
         @app.route('/predict_api',methods=['POST'])
         def predict_api():
             For direct API calls trought request
             data = request.get_json(force=True)
             prediction = model.predict([np.array(list(data.values()))])
             output = prediction
             return jsonify(output)
         if __name__ == "__main__":
             app.run(debug=True)
```

3. Module.py

finally used requests model.py to call APIs defined in app.py.