Name : Aggelos Konioris

Batch code : LISUM02

Submission date : 14/8/2021

Submitted to : Data Glacier

Short summary of our analysis

- Find the dataset of NBA players from the Kaggle source
- Use Python programming language to import the dataset and start the analysis
- Prepare the dataset using the suitable algorithms
- Utilize linear regression to predict the salary of an NBA player
- Save the model to our disk as pickle file and the load it to create the web application using Flask
- Write code on python for the web app
- Write HTML code on Notepad++ to complete our deployment on flask
- Use Anaconda prompt to run the python app and then attach the link with the http that is created to our browser to see our web app

Importing and organizing the dataset

```
In [1]: # Importing the dataset (Kaggle source) from our computer.
        import pandas as pd
        df = pd.read_csv('nba-stats-salary-rating.csv')
```

```
In [2]: # Dataset pre-processing step. We keep only the varibales which contain important information for our analysis.
        df['Salaries'] = df['Salaries'].str.replace('$','')
        df['Salaries'] = df['Salaries'].str.replace(',','')
        df['Salaries'] = pd.to_numeric(df['Salaries'])
        df.drop(['Unnamed: 0', 'Player', 'Tm', 'G', 'GS', 'ORB', 'DRB', 'FG', 'FGA', 'Pos', '3P', '3PA', '2P', '2PA', 'FT', 'FTA', 'eFG%
        df.rename(columns = {'MP':'Minutes_Played', 'FG%':'Fieldgoal_Percentage', '3P%':'Threepoint_Percentage', '2P%':'Twopoint_Percenta
        df['Fieldgoal_Percentage'] = df['Fieldgoal_Percentage']*100
        df['Threepoint_Percentage'] = df['Threepoint_Percentage']*100
        df['Twopoint_Percentage'] = df['Twopoint_Percentage']*100
        df['Freethrow_Percentage'] = df['Freethrow_Percentage']*100
        df
```

Splitting the dataset into train and test dataset then perform linear regression model and save the model to the disk

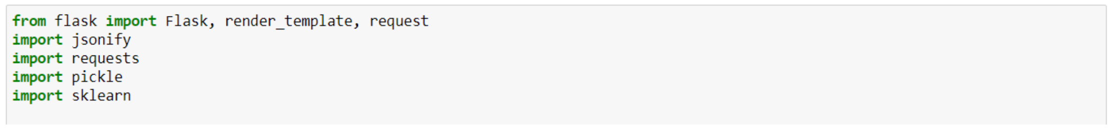
```
In [7]: # Splitting our dataset into train and test set in order to perfrom the Machine Learning model.
        from sklearn.model_selection import train_test_split
        X = df.drop('Salaries', axis = 1)
        y = df.iloc[:, 1]
        X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = .8)
```

```
In [8]: # We utilize a simple linear regression model
        from sklearn.linear_model import LinearRegression
        reg = LinearRegression()
        reg.fit(X_train, y_train)
        reg.predict(X_test)
```

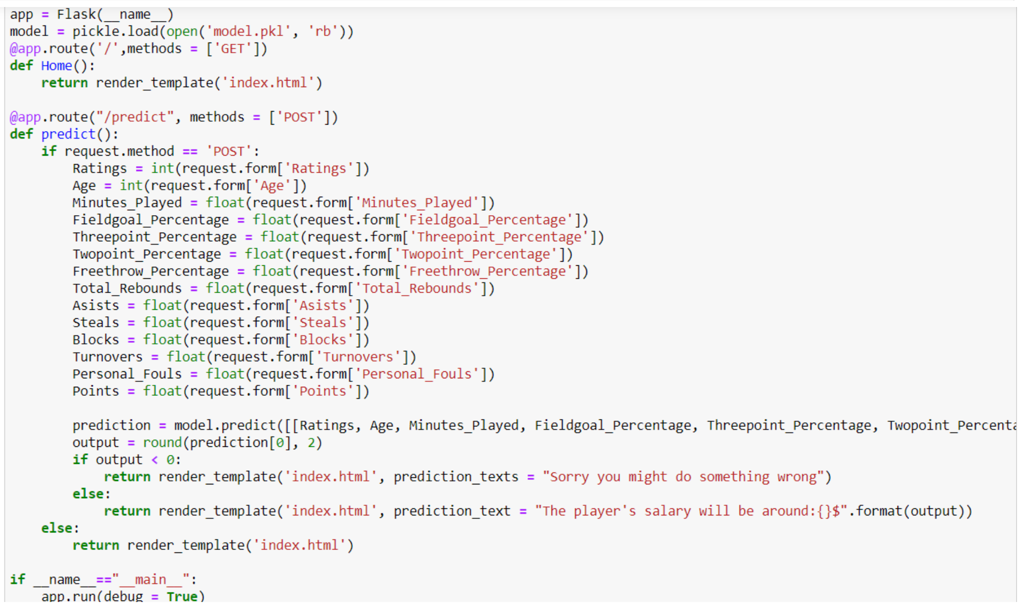
```
Out[8]: array([ 1758356.95916533, 12575635.46604124,  -134499.97848286,
                8226981.45375681, -2461982.52230784, 15925591.03062804,
               10224163.76179139,  3366232.94944562,  7581213.85396183,
               12962900.30026697,  7647406.02905659, 13199227.08827509,
                8691173.45685692, 12632269.62385233, -1179618.73801902,
               18672276.99618761, 13622838.0875452 ,  2308649.86353765,
               -2214833.11001204,  5105602.55294898,  5058144.42920403,
                7862697.34501329, 10438549.875103  , -1399107.82405333,
                  84174.0825462 , 12275431.66064784, 18533154.69019945,
                -387945.44150293,  8974493.2334785 ,  5809938.09754315,
                8433465.53759043,  -230512.96320127, 11985939.47052383,
                8962654.41636229,  7548710.10186614,  6584987.60455705,
               15009866.32879741,  6076361.14283407,  5730940.77535066,
                8534203.18861747,  6583541.1275066 ,  6170897.52757628,
               14749014.22672088,  8574817.45160988, 13029171.6560102 ,
               -1518652.19372076, 12882252.1863787 ,  7673608.52009942,
                2690442.02322255,  2271397.32747789,   449908.69996114,
               33030778.36726831,  -618500.57937242,  1191442.18920547,
               11989253.03231663,  4815307.59905493,  6126143.93499775,
                1498131.6704182 , -3068546.90657696,   553795.69405374,
                5832897.47858502,  7257362.10195972,  4013772.48853436,
                6776254.43212125,  1900258.2767535 , 13174778.15565005,
               12452872.15605174,  2469005.9274929 ,  3895913.35101463,
               10173694.19205971,  4412541.20978984,  4867847.75267954,
                6636579.65480135,  3044041.02044977, 16225134.66528565,
               15144946.98994382,  7353018.36996214, 18516647.95171857,
                2040528.35040373,  6534769.03970805, 21028772.60343842,
                8972994.74996349, 17186057.30750105,  -796285.83805937,
               -5307478.02792199,   828472.9806468 ,  8710358.89163746,
                2833253.6119372 , 14067797.82397592])
```

```
In [9]: # Save the model to disk.
        import pickle
        pickle.dump(reg, open('model.pkl', 'wb'))
```

Using Flask to write code for the deployment

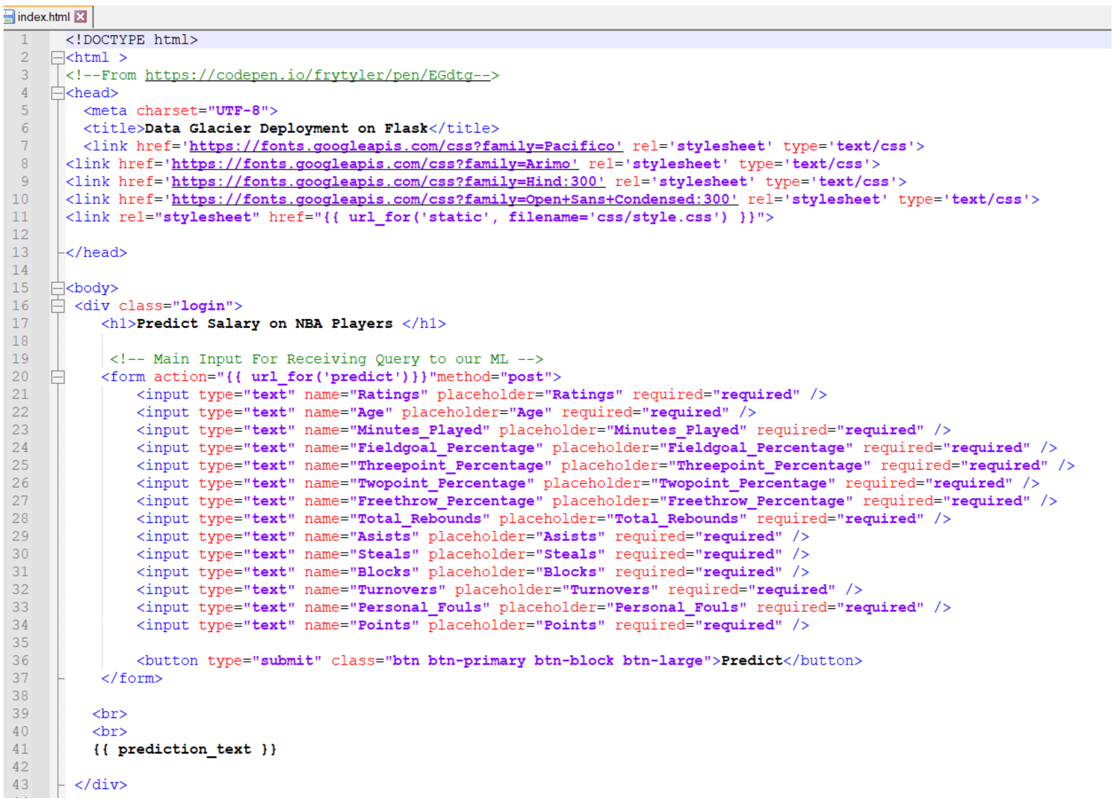


```python
from flask import Flask, render_template, request
import jsonify
import requests
import pickle
import sklearn
```



```python
app = Flask(__name__)
model = pickle.load(open('model.pkl', 'rb'))
@app.route('/',methods = ['GET'])
def Home():
    return render_template('index.html')

@app.route("/predict", methods = ['POST'])
def predict():
    if request.method == 'POST':
        Ratings = int(request.form['Ratings'])
        Age = int(request.form['Age'])
        Minutes_Played = float(request.form['Minutes_Played'])
        Fieldgoal_Percentage = float(request.form['Fieldgoal_Percentage'])
        Threepoint_Percentage = float(request.form['Threepoint_Percentage'])
        Twopoint_Percentage = float(request.form['Twopoint_Percentage'])
        Freethrow_Percentage = float(request.form['Freethrow_Percentage'])
        Total_Rebounds = float(request.form['Total_Rebounds'])
        Asists = float(request.form['Asists'])
        Steals = float(request.form['Steals'])
        Blocks = float(request.form['Blocks'])
        Turnovers = float(request.form['Turnovers'])
        Personal_Fouls = float(request.form['Personal_Fouls'])
        Points = float(request.form['Points'])

        prediction = model.predict([[Ratings, Age, Minutes_Played, Fieldgoal_Percentage, Threepoint_Percentage, Twopoint_Percenta
        output = round(prediction[0], 2)
        if output < 0:
            return render_template('index.html', prediction_texts = "Sorry you might do something wrong")
        else:
            return render_template('index.html', prediction_text = "The player's salary will be around:{}$".format(output))
    else:
        return render_template('index.html')

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

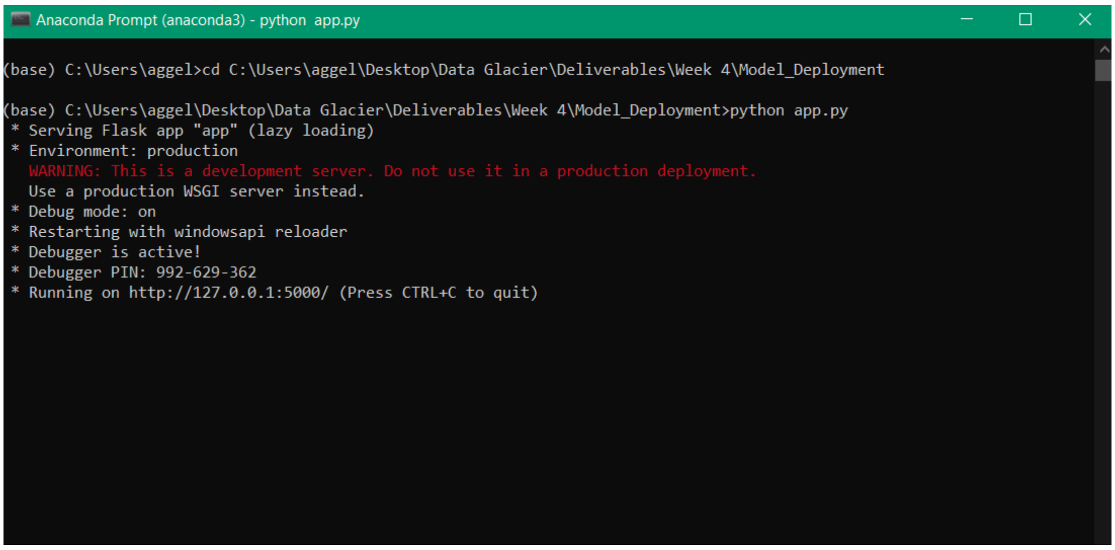
HTML code for the web application



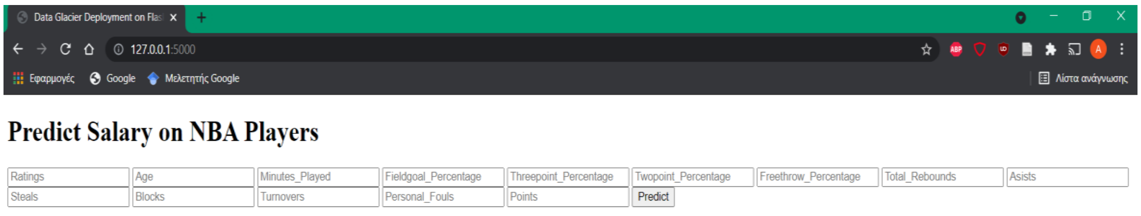
index.html

```html
<!DOCTYPE html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
  <meta charset="UTF-8">
  <title>Data Glacier Deployment on Flask</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">

</head>

<body>
 <div class="login">
    <h1>Predict Salary on NBA Players </h1>

     <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}"method="post">
        <input type="text" name="Ratings" placeholder="Ratings" required="required" />
        <input type="text" name="Age" placeholder="Age" required="required" />
        <input type="text" name="Minutes_Played" placeholder="Minutes_Played" required="required" />
        <input type="text" name="Fieldgoal_Percentage" placeholder="Fieldgoal_Percentage" required="required" />
        <input type="text" name="Threepoint_Percentage" placeholder="Threepoint_Percentage" required="required" />
        <input type="text" name="Twopoint_Percentage" placeholder="Twopoint_Percentage" required="required" />
        <input type="text" name="Freethrow_Percentage" placeholder="Freethrow_Percentage" required="required" />
        <input type="text" name="Total_Rebounds" placeholder="Total_Rebounds" required="required" />
        <input type="text" name="Asists" placeholder="Asists" required="required" />
        <input type="text" name="Steals" placeholder="Steals" required="required" />
        <input type="text" name="Blocks" placeholder="Blocks" required="required" />
        <input type="text" name="Turnovers" placeholder="Turnovers" required="required" />
        <input type="text" name="Personal_Fouls" placeholder="Personal_Fouls" required="required" />
        <input type="text" name="Points" placeholder="Points" required="required" />

        <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
    </form>

   <br>
   <br>
   {{ prediction_text }}

 </div>
```

Anaconda prompt with the http



```
(base) C:\Users\aggel>cd C:\Users\aggel\Desktop\Data Glacier\Deliverables\Week 4\Model_Deployment

(base) C:\Users\aggel\Desktop\Data Glacier\Deliverables\Week 4\Model_Deployment>python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with windowsapi reloader
 * Debugger is active!
 * Debugger PIN: 992-629-362
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```



Predict Salary on NBA Players

Ratings | Age | Minutes_Played | Fieldgoal_Percentage | Threepoint_Percentage | Twopoint_Percentage | Freethrow_Percentage | Total_Rebounds | Asists
Steals | Blocks | Turnovers | Personal_Fouls | Points | Predict