In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [2]:
```
data = pd.read_csv('C:\\Users\\anivi\\OneDrive\\Desktop\\placement.csv')
```

In [3]:
```
data.head()
```

Out[3]:

	cgpa	placement_exam_marks	placed
0	7.19	26.0	1
1	7.46	38.0	1
2	7.54	40.0	1
3	6.42	8.0	1
4	7.23	17.0	0

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 3 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   cgpa                  1000 non-null   float64
 1   placement_exam_marks  1000 non-null   float64
 2   placed                1000 non-null   int64  
dtypes: float64(2), int64(1)
memory usage: 23.6 KB
```

In [5]:
```
data.dtypes
```

Out[5]:
```
cgpa                    float64
placement_exam_marks    float64
placed                    int64
dtype: object
```

In [6]:
```
data.describe().T
```

Out[6]:

	count	mean	std	min	25%	50%	75%	max
cgpa	1000.0	6.96124	0.615898	4.89	6.55	6.96	7.37	9.12
placement_exam_marks	1000.0	32.22500	19.130822	0.00	17.00	28.00	44.00	100.00
placed	1000.0	0.48900	0.500129	0.00	0.00	0.00	1.00	1.00

In [7]:
```
data.columns
```

Out[7]:
```
Index(['cgpa', 'placement_exam_marks', 'placed'], dtype='object')
```

In [8]:
```
data.tail()
```

Out[8]:

	cgpa	placement_exam_marks	placed
995	8.87	44.0	1
996	9.12	65.0	1
997	4.89	34.0	0
998	8.62	46.0	1
999	4.90	10.0	1

In [9]:
```
data.isnull().sum()
```

Out[9]:
```
cgpa                    0
placement_exam_marks    0
placed                  0
dtype: int64
```

In [10]:
```
X = data[['cgpa', 'placement_exam_marks']]
y = data['placed']
```

In [12]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
train_score = model.score(X_train, y_train)
test_score = model.score(X_test, y_test)

print(f'Training R-squared: {train_score}')
print(f'Testing R-squared: {test_score}')
```

```
Training R-squared: 0.001382798269560781
Testing R-squared: -0.00377614302226581
```

In [13]:
```
import joblib
joblib.dump(model, 'model.pkl')
```

Out[13]:
```
['model.pkl']
```

In [14]:
```
!pip install flask-ngrok
```

```
Requirement already satisfied: flask-ngrok in c:\users\anivi\anaconda3\lib\site-packages (0.0.25)
Requirement already satisfied: Flask>=0.8 in c:\users\anivi\anaconda3\lib\site-packages (from flask-ngrok) (1.1.2)
Requirement already satisfied: requests in c:\users\anivi\anaconda3\lib\site-packages (from flask-ngrok) (2.28.1)
Requirement already satisfied: itsdangerous>=0.24 in c:\users\anivi\anaconda3\lib\site-packages (from Flask>=0.8->flask-ngrok) (2.0.1)
Requirement already satisfied: click>=5.1 in c:\users\anivi\anaconda3\lib\site-packages (from Flask>=0.8->flask-ngrok) (8.0.4)
Requirement already satisfied: Jinja2>=2.10.1 in c:\users\anivi\anaconda3\lib\site-packages (from Flask>=0.8->flask-ngrok) (2.11.3)
Requirement already satisfied: Werkzeug>=0.15 in c:\users\anivi\anaconda3\lib\site-packages (from Flask>=0.8->flask-ngrok) (2.0.3)
Requirement already satisfied: idna<4,>=2.5 in c:\users\anivi\anaconda3\lib\site-packages (from requests->flask-ngrok) (3.3)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\anivi\anaconda3\lib\site-packages (from requests->flask-ngrok) (1.26.11)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\anivi\anaconda3\lib\site-packages (from requests->flask-ngrok) (2022.9.14)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\anivi\anaconda3\lib\site-packages (from requests->flask-ngrok) (2.0.4)
Requirement already satisfied: colorama in c:\users\anivi\anaconda3\lib\site-packages (from click>=5.1->Flask>=0.8->flask-ngrok) (0.4.5)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\anivi\anaconda3\lib\site-packages (from Jinja2>=2.10.1->Flask>=0.8->flask-ngrok) (2.0.1)
```

In [ ]:
```
from flask import Flask, request, jsonify
import joblib
from flask_ngrok import run_with_ngrok

app = Flask(__name__)
run_with_ngrok(app)  # Start ngrok when the app is run

# Load the pre-trained model
model = joblib.load('model.pkl')

# Define API endpoint for model prediction
@app.route('/predict', methods=['GET'])
def predict():
    try:
        # Parse input data
        data = request.get_json()
        feature1 = data['feature1']
        feature2 = data['feature2']

        # Make predictions using the loaded model
        prediction = model.predict([[feature1, feature2]])

        # Return predictions
        return jsonify({'prediction': int(prediction[0])})

    except Exception as e:
        return jsonify({'error': str(e)})

# Run the Flask app
if __name__ == '__main__':
    app.run()
```

```
 * Serving Flask app "__main__" (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: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Running on http://800f-86-12-242-125.ngrok.io
 * Traffic stats available on http://127.0.0.1:4040
127.0.0.1 - - [29/Jul/2023 18:35:40] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [29/Jul/2023 18:35:41] "GET /static/EuclidSquare-Medium-WebS.woff HTTP/1.1" 404 -
127.0.0.1 - - [29/Jul/2023 18:35:41] "GET /static/EuclidSquare-Regular-WebS.woff HTTP/1.1" 404 -
127.0.0.1 - - [29/Jul/2023 18:35:41] "GET /static/IBMPlexMono-SemiBoldItalic.woff HTTP/1.1" 404 -
127.0.0.1 - - [29/Jul/2023 18:35:41] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [29/Jul/2023 18:35:42] "GET /static/IBMPlexMono-TextItalic.woff HTTP/1.1" 404 -
127.0.0.1 - - [29/Jul/2023 18:35:42] "GET /static/IBMPlexMono-SemiBold.woff HTTP/1.1" 404 -
127.0.0.1 - - [29/Jul/2023 18:35:42] "GET /static/IBMPlexMono-Text.woff HTTP/1.1" 404 -
127.0.0.1 - - [29/Jul/2023 18:35:42] "GET /static/EuclidSquare-MediumItalic-WebS.woff HTTP/1.1" 404 -
127.0.0.1 - - [29/Jul/2023 18:35:42] "GET /static/EuclidSquare-RegularItalic-WebS.woff HTTP/1.1" 404 -
127.0.0.1 - - [29/Jul/2023 18:35:54] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [29/Jul/2023 18:35:54] "GET /favicon.ico HTTP/1.1" 404 -
```

In [ ]:

Name: Anivirudhan Ramesh

Batch Code: LISUM23

Submission Date: 28/07/2023

Submitted to : Data Glacier

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [2]:
```
data = pd.read_csv('C:\\Users\\anivi\\OneDrive\\Desktop\\placement.csv')
```

In [3]:
```
data.head()
```

Out[3]:

	cgpa	placement_exam_marks	placed
0	7.19	26.0	1
1	7.46	38.0	1
2	7.54	40.0	1
3	6.42	8.0	1
4	7.23	17.0	0

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 3 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   cgpa                  1000 non-null   float64
 1   placement_exam_marks  1000 non-null   float64
 2   placed                1000 non-null   int64  
dtypes: float64(2), int64(1)
memory usage: 23.6 KB
```

In [5]:
```
data.dtypes
```

Out[5]:
```
cgpa                    float64
placement_exam_marks    float64
placed                    int64
dtype: object
```

In [6]:
```
data.describe().T
```

Out[6]:

	count	mean	std	min	25%	50%	75%	max
cgpa	1000.0	6.96124	0.615898	4.89	6.55	6.96	7.37	9.12
placement_exam_marks	1000.0	32.22500	19.130822	0.00	17.00	28.00	44.00	100.00
placed	1000.0	0.48900	0.500129	0.00	0.00	0.00	1.00	1.00

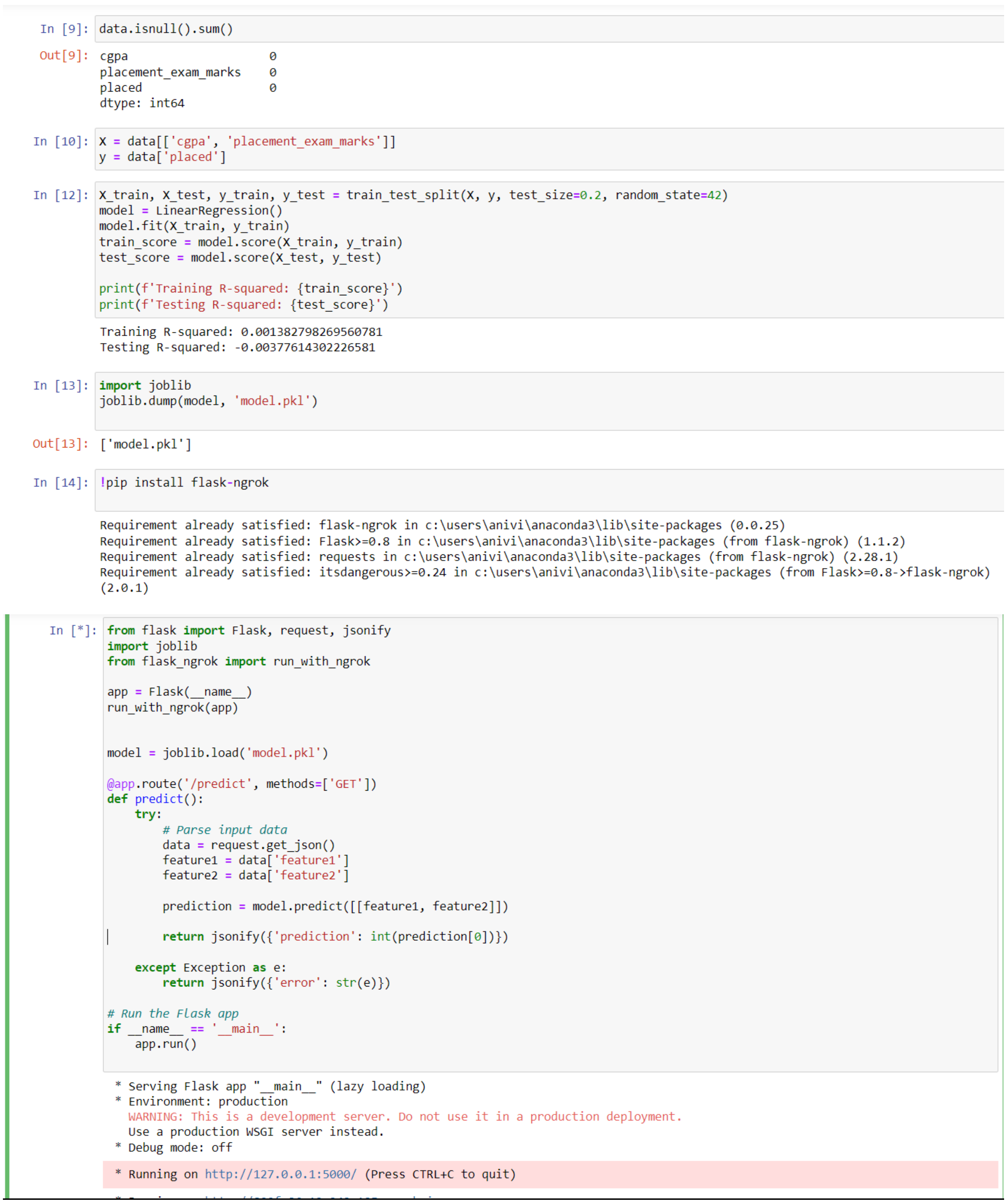
In [7]:
```
data.columns
```

Out[7]:
```
Index(['cgpa', 'placement_exam_marks', 'placed'], dtype='object')
```

In [8]:
```
data.tail()
```

Out[8]:

	cgpa	placement_exam_marks	placed
995	8.87	44.0	1
996	9.12	65.0	1
997	4.89	34.0	0
998	8.62	46.0	1
999	4.90	10.0	1



```
In [9]: data.isnull().sum()

Out[9]: cgpa                   0
        placement_exam_marks   0
        placed                 0
        dtype: int64
```

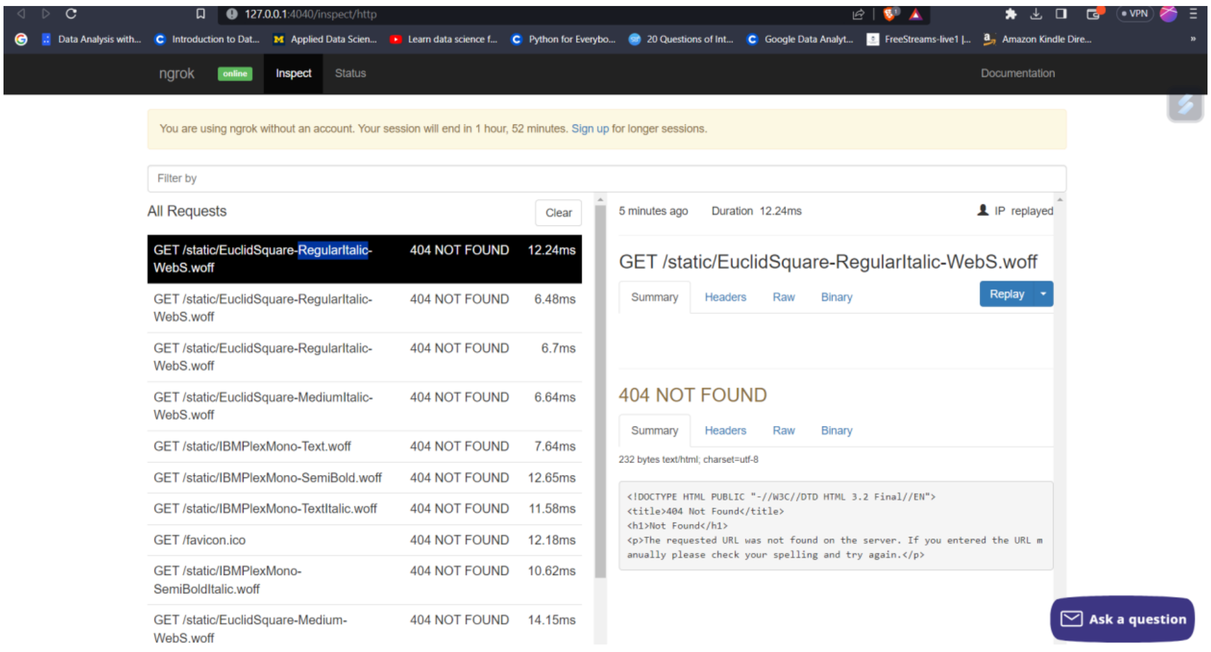
```
In [10]: X = data[['cgpa', 'placement_exam_marks']]
         y = data['placed']
```

```
In [12]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
         model = LinearRegression()
         model.fit(X_train, y_train)
         train_score = model.score(X_train, y_train)
         test_score = model.score(X_test, y_test)

         print(f'Training R-squared: {train_score}')
         print(f'Testing R-squared: {test_score}')

         Training R-squared: 0.001382798269560781
         Testing R-squared: -0.00377614302226581
```

```
In [13]: import joblib
         joblib.dump(model, 'model.pkl')

Out[13]: ['model.pkl']
```

```
In [14]: !pip install flask-ngrok

         Requirement already satisfied: flask-ngrok in c:\users\anivi\anaconda3\lib\site-packages (0.0.25)
         Requirement already satisfied: Flask>=0.8 in c:\users\anivi\anaconda3\lib\site-packages (from flask-ngrok) (1.1.2)
         Requirement already satisfied: requests in c:\users\anivi\anaconda3\lib\site-packages (from flask-ngrok) (2.28.1)
         Requirement already satisfied: itsdangerous>=0.24 in c:\users\anivi\anaconda3\lib\site-packages (from Flask>=0.8->flask-ngrok)
         (2.0.1)
```

```
In [*]: from flask import Flask, request, jsonify
        import joblib
        from flask_ngrok import run_with_ngrok

        app = Flask(__name__)
        run_with_ngrok(app)


        model = joblib.load('model.pkl')

        @app.route('/predict', methods=['GET'])
        def predict():
            try:
                # Parse input data
                data = request.get_json()
                feature1 = data['feature1']
                feature2 = data['feature2']

                prediction = model.predict([[feature1, feature2]])

                return jsonify({'prediction': int(prediction[0])})

            except Exception as e:
                return jsonify({'error': str(e)})

        # Run the Flask app
        if __name__ == '__main__':
            app.run()

         * Serving Flask app "__main__" (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: off
         * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```



You are using ngrok without an account. Your session will end in 1 hour, 52 minutes. Sign up for longer sessions.

Filter by

All Requests

GET /static/EuclidSquare-RegularItalic-WebS.woff	404 NOT FOUND	12.24ms
GET /static/EuclidSquare-RegularItalic-WebS.woff	404 NOT FOUND	6.48ms
GET /static/EuclidSquare-RegularItalic-WebS.woff	404 NOT FOUND	6.7ms
GET /static/EuclidSquare-MediumItalic-WebS.woff	404 NOT FOUND	6.64ms
GET /static/IBMPlexMono-Text.woff	404 NOT FOUND	7.64ms
GET /static/IBMPlexMono-SemiBold.woff	404 NOT FOUND	12.65ms
GET /static/IBMPlexMono-TextItalic.woff	404 NOT FOUND	11.58ms
GET /favicon.ico	404 NOT FOUND	12.18ms
GET /static/IBMPlexMono-SemiBoldItalic.woff	404 NOT FOUND	10.62ms
GET /static/EuclidSquare-Medium-WebS.woff	404 NOT FOUND	14.15ms

5 minutes ago Duration 12.24ms IP replayed

GET /static/EuclidSquare-RegularItalic-WebS.woff

Summary Headers Raw Binary

404 NOT FOUND

Summary Headers Raw Binary

232 bytes text/html; charset=utf-8

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>404 Not Found</title>
<h1>Not Found</h1>
<p>The requested URL was not found on the server. If you entered the URL m
anually please check your spelling and try again.</p>
```