

NAME: BOLUTIFE AKINLAWON

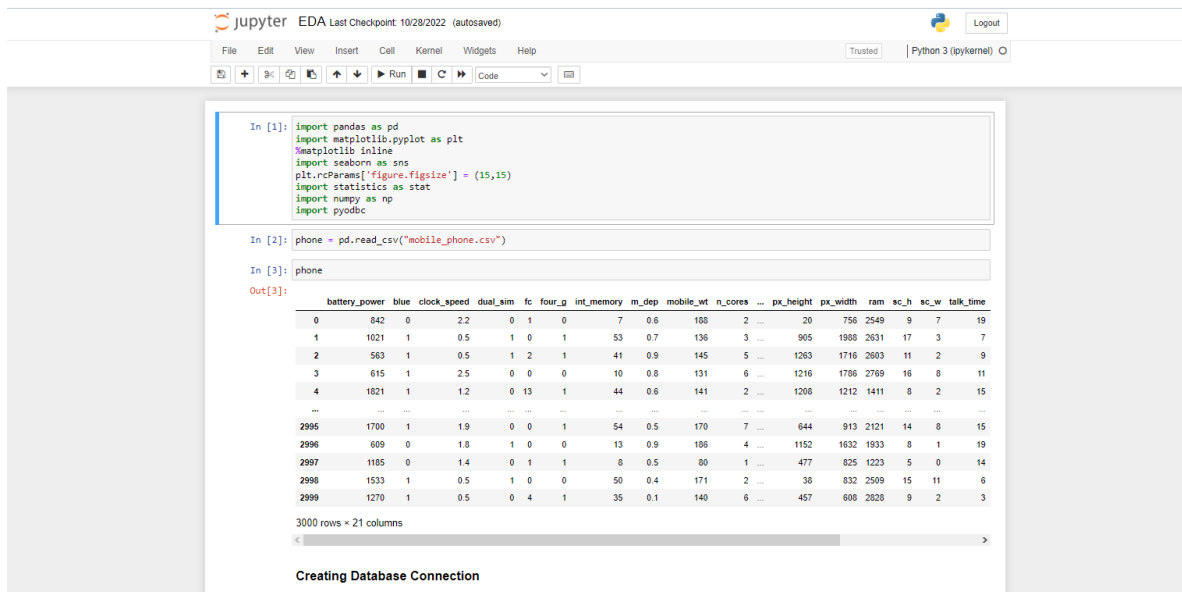
BATCH CODE: LISUM14

SUBMISSION DATE: 10/11/2022

SUBMITTED TO: GITHUB

FLASK DEPLOYMENT

Importing required Libraries and data preparation



The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [1]: import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
plt.rcParams['figure.figsize'] = (15,15)
import statistics as stat
import numpy as np
import pyodbc
```

```
In [2]: phone = pd.read_csv("mobile_phone.csv")
```

```
In [3]: phone
```

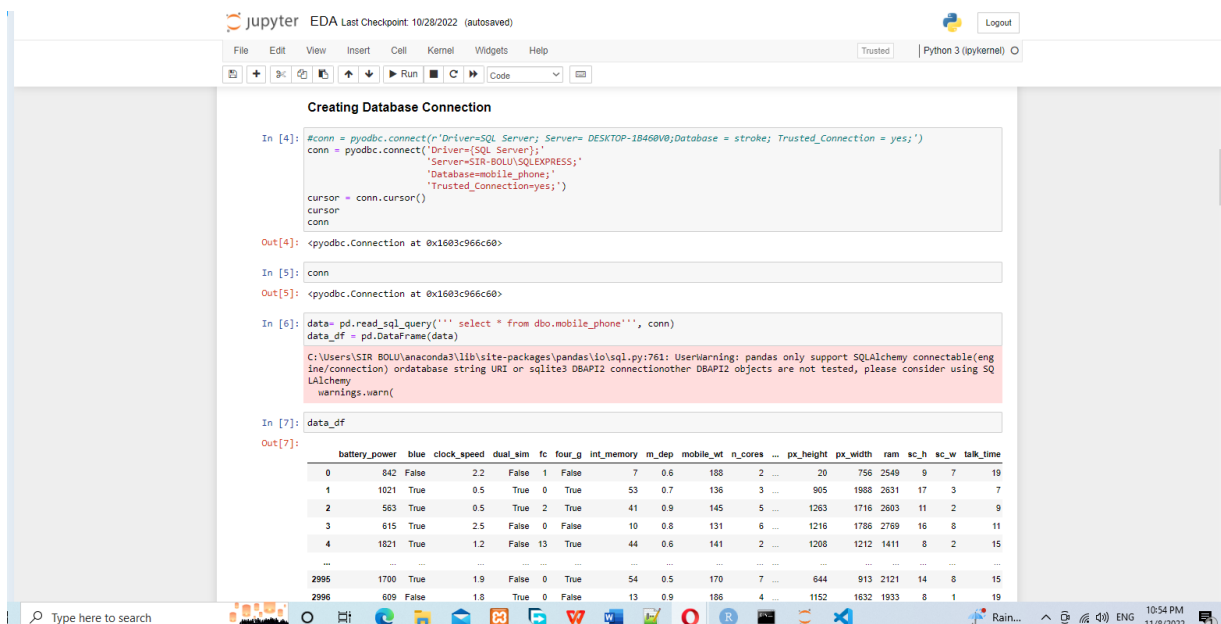
Out[3]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9	7	19
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	17	3	7
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	2	9
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	16	8	11
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1206	1212	1411	8	2	15
...
2995	1700	1	1.9	0	0	1	54	0.5	170	7	...	644	913	2121	14	8	15
2996	609	0	1.8	1	0	0	13	0.9	186	4	...	1152	1632	1933	8	1	19
2997	1185	0	1.4	0	1	1	8	0.5	80	1	...	477	825	1223	5	0	14
2998	1533	1	0.5	1	0	0	50	0.4	171	2	...	38	832	2509	15	11	6
2999	1270	1	0.5	0	4	1	35	0.1	140	6	...	457	608	2828	9	2	3

3000 rows x 21 columns

Creating Database Connection

Connecting to already created database and reading the dataframe



The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [4]: !conn = pyodbc.connect('Driver=SQL Server; Server=DESKTOP-1B460V0;Database = stroke; Trusted_Connection = yes;')
conn = pyodbc.connect('Driver=SQL Server;
                    Server=SIR-BOLU\SQLEXPRESS;
                    Database=mobile_phone;
                    Trusted_Connection=yes;')
```

```
Out[4]: <pyodbc.Connection at 0x1603c966c60>
```

```
In [5]: conn
```

```
Out[5]: <pyodbc.Connection at 0x1603c966c60>
```

```
In [6]: data = pd.read_sql_query('select * from dbo.mobile_phone', conn)
data_df = pd.DataFrame(data)
```

C:\Users\SIR-BOLU\anaconda3\lib\site-packages\pandas\io\sql.py:761: UserWarning: pandas only support SQLAlchemy connectable(engine/connection) or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested, please consider using SQLAlchemy warnings.warn()

```
In [7]: data_df
```

Out[7]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time
0	842	False	2.2	False	1	False	7	0.6	188	2	...	20	756	2549	9	7	19
1	1021	True	0.5	True	0	True	53	0.7	136	3	...	905	1988	2631	17	3	7
2	563	True	0.5	True	2	True	41	0.9	145	5	...	1263	1716	2603	11	2	9
3	615	True	2.5	False	0	False	10	0.8	131	6	...	1216	1786	2769	16	8	11
4	1821	True	1.2	False	13	True	44	0.6	141	2	...	1206	1212	1411	8	2	15
...
2995	1700	True	1.9	False	0	True	54	0.5	170	7	...	644	913	2121	14	8	15
2996	609	False	1.8	True	0	False	13	0.9	186	4	...	1152	1632	1933	8	1	19

Exploratory data analysis using datasist library

Jupyter EDA Last Checkpoint: 10/28/2022 (autosaved) Python 3 (ipykernel)

File Edit View Insert Cell Kernel Widgets Help

Run

Exploratory data Analysis using Datasist Library

```
In [9]: import datasist as ds

In [10]: #ds.structdata.describe(data_df)

In [11]: data_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 21 columns):
 #   Column              Non-Null Count  Dtype
---  --
 0   battery_power       3000 non-null   int64
 1   blue                3000 non-null   bool
 2   clock_speed         3000 non-null   float64
 3   dual_sim            3000 non-null   bool
 4   fc                  3000 non-null   int64
 5   four_g              3000 non-null   bool
 6   int_memory          3000 non-null   int64
 7   m_dep               3000 non-null   float64
 8   mobile_wt           3000 non-null   int64
 9   n_cores             3000 non-null   int64
10   pc                  3000 non-null   int64
11   px_height            3000 non-null   int64
12   px_width            3000 non-null   int64
13   ram                 3000 non-null   int64
14   sc_h                3000 non-null   int64
15   sc_w                3000 non-null   int64
16   talk_time           3000 non-null   int64
17   three_g             3000 non-null   bool
18   touch_screen        3000 non-null   bool
19   wifi                3000 non-null   bool
20   price_range         3000 non-null   int64
dtypes: bool(6), float64(2), int64(13)
memory usage: 369.3 KB
```

Jupyter EDA Last Checkpoint: 10/28/2022 (autosaved) Python 3 (ipykernel)

File Edit View Insert Cell Kernel Widgets Help

Run

```
wifi                bool
price_range         int64
dtype: object

In [22]: #correlation matrix
data_df.corr()

Out[22]:
```

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	px_height	px_width
battery_power	1.000000	-0.007606	-0.005250	-0.048135	0.019819	-0.003877	-0.000998	0.020382	-0.014217	-0.011659	0.025628	0.012056
blue	-0.007606	1.000000	0.026114	0.019910	-0.016035	0.007922	0.024133	0.009229	0.001789	0.024396	-0.024263	-0.030579
clock_speed	-0.005250	0.026114	1.000000	-0.004980	0.003514	-0.037196	-0.005467	-0.003707	0.003424	-0.008339	-0.004192	0.017658
dual_sim	-0.048135	0.019910	-0.004980	1.000000	0.000526	0.010192	-0.014199	-0.007585	-0.006660	0.017750	-0.011925	0.014643
fc	0.019819	-0.016035	0.003514	0.000526	1.000000	-0.000779	-0.020176	0.006548	0.021532	-0.002960	-0.013234	0.006695
four_g	-0.003877	0.007922	-0.037196	0.010192	-0.000779	1.000000	-0.008044	0.002775	-0.010920	0.003732	-0.001293	0.017684
int_memory	-0.000998	0.024133	-0.005467	-0.014199	-0.020176	-0.008044	1.000000	0.004277	-0.026789	-0.013321	0.003081	-0.007357
m_dep	0.020382	0.009229	-0.003707	-0.007585	0.006548	0.002775	0.004277	1.000000	0.000849	-0.000093	0.036750	0.026959
mobile_wt	-0.014217	0.001789	0.003424	-0.006660	0.021532	-0.010920	-0.026789	0.000849	1.000000	-0.025144	0.004442	-0.004675
n_cores	-0.011659	0.024396	-0.008339	0.017750	-0.002960	0.003732	-0.013321	-0.000093	-0.025144	1.000000	-0.021674	-0.003281
pc	0.025401	-0.014850	0.012682	0.013381	0.049565	0.008524	-0.014118	0.022075	0.021547	0.003587	-0.003090	0.021710
px_height	0.025628	-0.024263	-0.004192	-0.011925	-0.013234	-0.001293	0.003081	0.036750	0.004442	-0.021674	1.000000	0.012989
px_width	0.012056	-0.030579	0.017658	0.014643	0.006695	0.017684	-0.007357	0.026959	-0.004675	-0.003281	0.012989	1.000000
ram	-0.011059	0.036893	0.002139	0.043486	-0.007520	0.014950	0.019742	-0.000154	0.007720	-0.011279	-0.004612	-0.006271
sc_h	-0.038917	0.001891	-0.032997	-0.006008	0.007307	0.013950	0.020374	-0.026484	-0.029531	-0.010382	0.044085	0.006884
sc_w	-0.022727	0.000818	-0.014421	-0.012221	-0.007697	0.026152	0.013828	-0.021243	-0.006372	0.023479	0.044070	0.025355
talk_time	0.040337	-0.001305	-0.034143	-0.024706	-0.021831	-0.026627	0.006348	0.019500	-0.003097	0.006603	0.009971	0.022441
three_g	0.018081	-0.015693	-0.038021	-0.009127	-0.002797	0.573829	-0.011804	-0.017879	0.003172	0.007503	-0.024442	0.006894
touch_screen	-0.010421	-0.013350	0.033917	-0.000096	-0.004628	0.007922	-0.010725	0.011327	0.005084	0.010416	-0.008331	-0.014521
wifi	-0.005728	-0.006057	-0.032594	0.025674	-0.007223	-0.023620	0.008606	-0.032056	0.022733	-0.009054	0.030744	-0.004845
price_range	0.148885	0.023079	0.001212	0.009600	0.030507	0.011529	0.011998	0.010501	-0.020176	-0.004663	0.121875	0.108736

21 rows x 13 columns

ML Modelling with Random Forest

```
jupyter EDA Last Checkpoint: 10/28/2022 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [23]: data_df.columns
Out[23]: Index(['battery_power', 'blue', 'clock_speed', 'dual_sim', 'fc', 'four_g',
               'int_memory', 'm_dep', 'mobile_wt', 'n_cores', 'pc', 'px_height',
               'px_width', 'ram', 'sc_h', 'sc_w', 'talk_time', 'three_g',
               'touch_screen', 'wifi', 'price_range'],
              dtype='object')

In [24]: X = data_df.drop('price_range', axis=1)
         y = data_df.price_range

In [25]: from sklearn.model_selection import train_test_split
         x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

In [26]: from sklearn.ensemble import RandomForestRegressor

In [27]: model = RandomForestRegressor()
         model.fit(x_train, y_train)
Out[27]: RandomForestRegressor()

In [28]: predictions = model.predict(x_test)

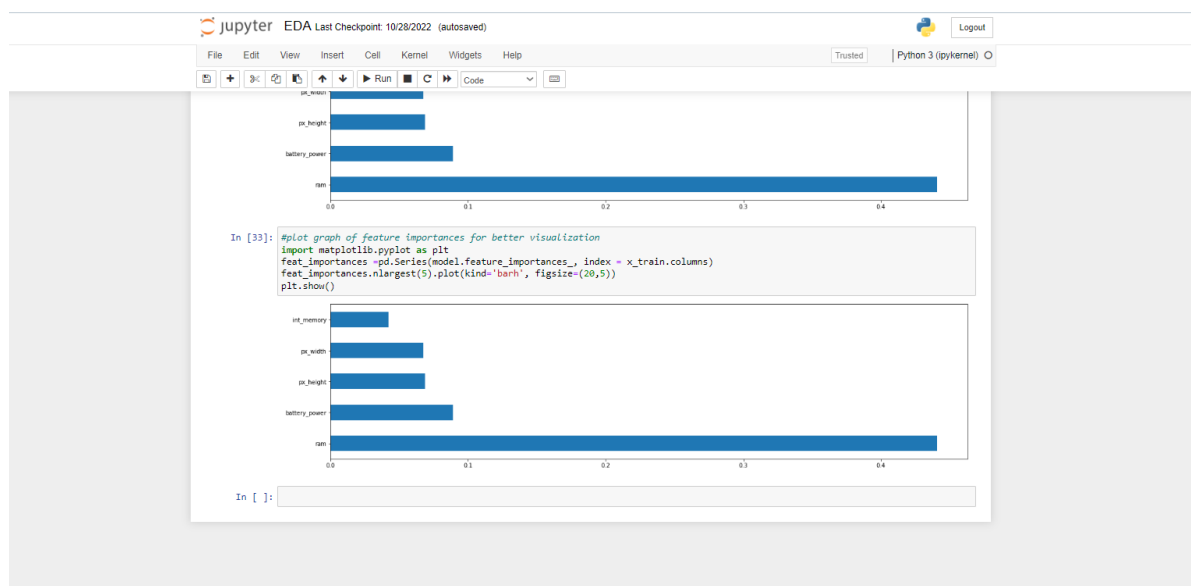
In [29]: from sklearn import metrics

In [30]: import numpy as np
         print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
RMSE: 0.8700865857296426

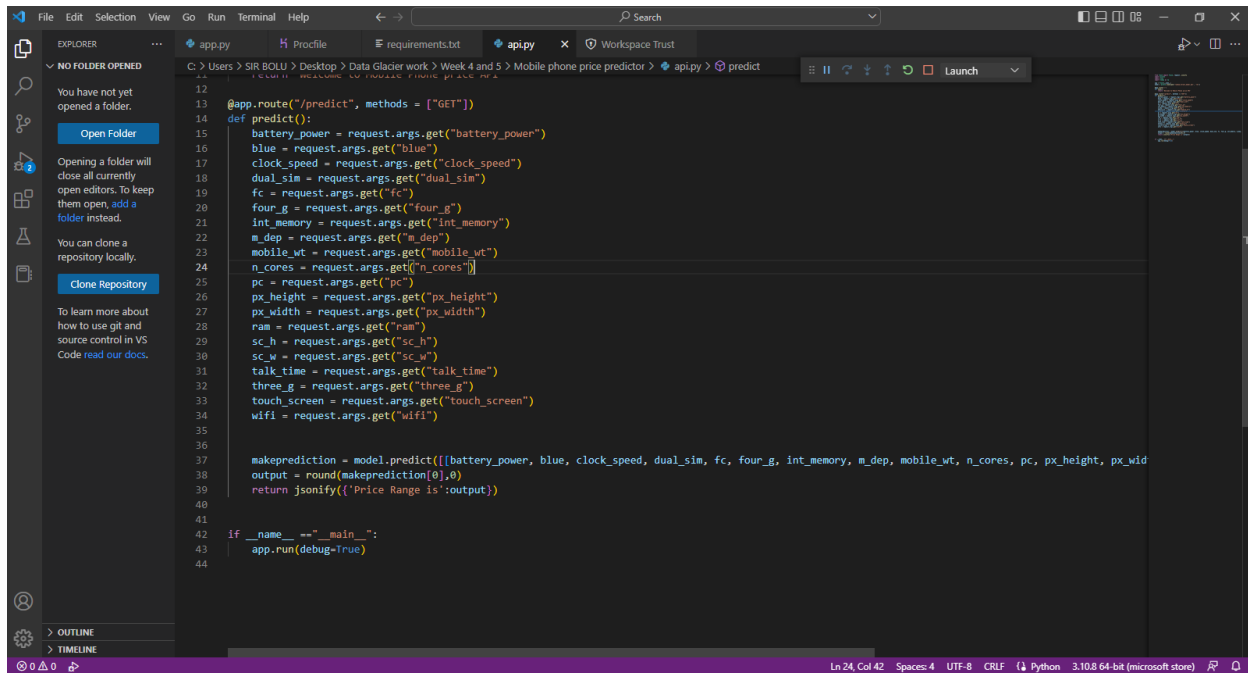
In [34]: import pickle
         file = open('random_forest_model.pkl', 'wb')
         pickle.dump(model, file)

In [32]: #Plot graph of feature importances for better visualization
         import matplotlib.pyplot as plt
```

Feature importance graph



API development with Flask



The screenshot shows the Visual Studio Code interface with a Python file named `app.py` open. The code is a Flask application that uses a pre-trained model to predict mobile phone prices based on various specifications. The left sidebar shows the Explorer view with a message about opening a folder. The bottom status bar indicates the current line and column (Ln 24, Col 42) and the Python environment (3.10.8 64-bit (microsoft store)).

```
11
12
13 @app.route("/predict", methods = ["GET"])
14 def predict():
15     battery_power = request.args.get("battery_power")
16     blue = request.args.get("blue")
17     clock_speed = request.args.get("clock_speed")
18     dual_sim = request.args.get("dual_sim")
19     fc = request.args.get("fc")
20     four_g = request.args.get("four_g")
21     int_memory = request.args.get("int_memory")
22     m_dep = request.args.get("m_dep")
23     mobile_wt = request.args.get("mobile_wt")
24     n_cores = request.args.get("n_cores")
25     pc = request.args.get("pc")
26     px_height = request.args.get("px_height")
27     px_width = request.args.get("px_width")
28     ram = request.args.get("ram")
29     sc_h = request.args.get("sc_h")
30     sc_w = request.args.get("sc_w")
31     talk_time = request.args.get("talk_time")
32     three_g = request.args.get("three_g")
33     touch_screen = request.args.get("touch_screen")
34     wifi = request.args.get("wifi")
35
36
37     makeprediction = model.predict([[battery_power, blue, clock_speed, dual_sim, fc, four_g, int_memory, m_dep, mobile_wt, n_cores, pc, px_height, px_wid
38     output = round(makeprediction[0],0)
39     return jsonify({'Price Range is':output})
40
41
42 if __name__ == "__main__":
43     app.run(debug=True)
44
```