In lab Task:`

Code output:

```
[16] !wget https://www.dropbox.com/s/veak3ugc4wj9luz/Alumni%20Giving%20Regression%20%28Edited%29.csv?dl=0 -O -quiet "./Alumni Giving Regression (Edite

--2023-10-19 10:52:21--  https://www.dropbox.com/s/veak3ugc4wj9luz/Alumni%20Giving%20Regression%20%28Edited%29.csv?dl=0
Resolving www.dropbox.com (www.dropbox.com)... 162.125.67.18, 2620:100:601f:18::a27d:912
Connecting to www.dropbox.com (www.dropbox.com)|162.125.67.18|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: /s/raw/veak3ugc4wj9luz/Alumni%20Giving%20Regression%20%28Edited%29.csv [following]
--2023-10-19 10:52:21--  https://www.dropbox.com/s/raw/veak3ugc4wj9luz/Alumni%20Giving%20Regression%20%28Edited%29.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uce78cfd066a315bcd59d926431c.dl.dropboxusercontent.com/cd/0/inline/CF6rhm_X2oht9bBirZE6rbjvJKBpzIaD5yi-D8XJjy_XlVg7gBzJf68wZrww1
--2023-10-19 10:52:22--  https://uce78cfd066a315bcd59d926431c.dl.dropboxusercontent.com/cd/0/inline/CF6rhm_X2oht9bBirZE6rbjvJKBpzIaD5yi-D8XJjy_XlV
Resolving uce78cfd066a315bcd59d926431c.dl.dropboxusercontent.com (uce78cfd066a315bcd59d926431c.dl.dropboxusercontent.com)... 162.125.2.15, 2620:10
Connecting to uce78cfd066a315bcd59d926431c.dl.dropboxusercontent.com (uce78cfd066a315bcd59d926431c.dl.dropboxusercontent.com)|162.125.2.15|:443...
HTTP request sent, awaiting response... 200 OK
Length: 3504 (3.4K) [text/plain]
Saving to: '-quiet'

-quiet              100%[==================>]   3.42K  --.-KB/s    in 0s

2023-10-19 10:52:22 (1.01 GB/s) - '-quiet' saved [3504/3504]

--2023-10-19 10:52:22--  http://./Alumni%20Giving%20Regression%20(Edited).csv
Resolving . (.)... failed: No address associated with hostname.
wget: unable to resolve host address '.'
FINISHED --2023-10-19 10:52:22--
Total wall clock time: 1.7s
Downloaded: 1 files, 3.4K in 0s (1.01 GB/s)
```

```
from keras.models import Sequential
from keras.layers import Dense, Dropout
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import numpy as np
from sklearn import linear_model
from sklearn import preprocessing
from sklearn import tree
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
import pandas as pd
import csv
```

```
[8] np.random.seed(7)
    df = pd.read_csv("Alumni Giving Regression (Edited).csv", delimiter="," )
    df.head()
```

	A	B	C	D	E	F
0	24	0.42	0.16	0.59	0.81	0.08
1	19	0.49	0.04	0.37	0.69	0.11
2	18	0.24	0.17	0.66	0.87	0.31
3	8	0.74	0.00	0.81	0.88	0.11
4	8	0.95	0.00	0.86	0.92	0.28

```
df.describe()
```

	A	B	C	D	E	F
count	123.000000	123.000000	123.000000	123.000000	123.000000	123.000000
mean	17.772358	0.403659	0.136260	0.645203	0.841138	0.141789
std	4.517385	0.133897	0.060101	0.169794	0.083942	0.080674
min	6.000000	0.140000	0.000000	0.260000	0.580000	0.020000
25%	16.000000	0.320000	0.095000	0.505000	0.780000	0.080000
50%	18.000000	0.380000	0.130000	0.640000	0.840000	0.130000
75%	20.000000	0.460000	0.180000	0.785000	0.910000	0.170000
max	31.000000	0.950000	0.310000	0.960000	0.980000	0.410000

```
[10] corr=df.corr(method ='pearson')
     corr
```

	A	B	C	D	E	F
A	1.000000	-0.691900	0.414978	-0.604574	-0.521985	-0.549244
B	-0.691900	1.000000	-0.581516	0.487248	0.376735	0.540427
C	0.414978	-0.581516	1.000000	0.017023	0.055766	-0.175102
D	-0.604574	0.487248	0.017023	1.000000	0.934396	0.681660
E	-0.521985	0.376735	0.055766	0.934396	1.000000	0.647625
F	-0.549244	0.540427	-0.175102	0.681660	0.647625	1.000000

```
[11] Y_POSITION = 5
     model_1_features = [i for i in range(0,Y_POSITION)]
     X = df.iloc[:,model_1_features]
     Y = df.iloc[:,Y_POSITION]
     X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.20, random_state=2020)
```

```
[15] model1 = linear_model.LinearRegression()
     model1.fit(X_train, y_train)
     y_pred_train1 = model1.predict(X_train)
     print("Regression")
     print("================================")
     RMSE_train1 = mean_squared_error(y_train,y_pred_train1)
     print("Regression Train set: RMSE {}".format(RMSE_train1))
     print("================================")
     y_pred1 = model1.predict(X_test)
     RMSE_test1 = mean_squared_error(y_test,y_pred1)
     print("Regression Test set: RMSE {}".format(RMSE_test1))
     print("================================")
     coef_dict = {}
     for coef, feat in zip(model1.coef_,model_1_features):
       coef_dict[df.columns[feat]] = coef
     print(coef_dict)
```

```
Regression
================================
Regression Train set: RMSE 0.002761693322289229
================================
Regression Test set: RMSE 0.004209824026356377
================================
{'A': -0.0009337757382416938, 'B': 0.16012156890162943, 'C': -0.044160015425349614, 'D': 0.15217907817100407, 'E': 0.17539950794101047}
```

Why Keras Sequential model?

There are cases where deep learning models, including sequential models, can outperform classical machine learning methods like linear regression. These cases typically involve very large and complex datasets, and problems where the relationship between inputs and outputs is highly non-linear and difficult to model with traditional methods. Hence sequential model is not the best use for linear regression.

What other models are available and what are their differences?

Traditional Machine Learning Models:

1. **Sequential Model**: This is the simplest type of model in Keras, where you can stack layers sequentially. It's commonly used for feedforward neural networks.

Pre-Trained Deep Learning Models:

2. **VGG16 and VGG19**: Deep convolutional neural networks widely used for image classification tasks.
3. **ResNet**: Residual Networks are known for handling very deep architectures effectively. Variants include ResNet50, ResNet101, etc.
4. **InceptionV3**: Known for its performance on image classification tasks. It uses multiple parallel paths for processing images.

Sequential models in Keras allow for custom neural network architectures, enabling flexibility in design and training from scratch. They are versatile for various tasks. Pre-trained models, on the other hand, are pre-built architectures optimized for specific tasks like image classification. They're resource-intensive but excel in transfer learning, where they can be fine-tuned on new datasets. Primarily used for computer vision tasks, they offer powerful features for tasks like image classification and object detection. The choice depends on the specific task and resources available.