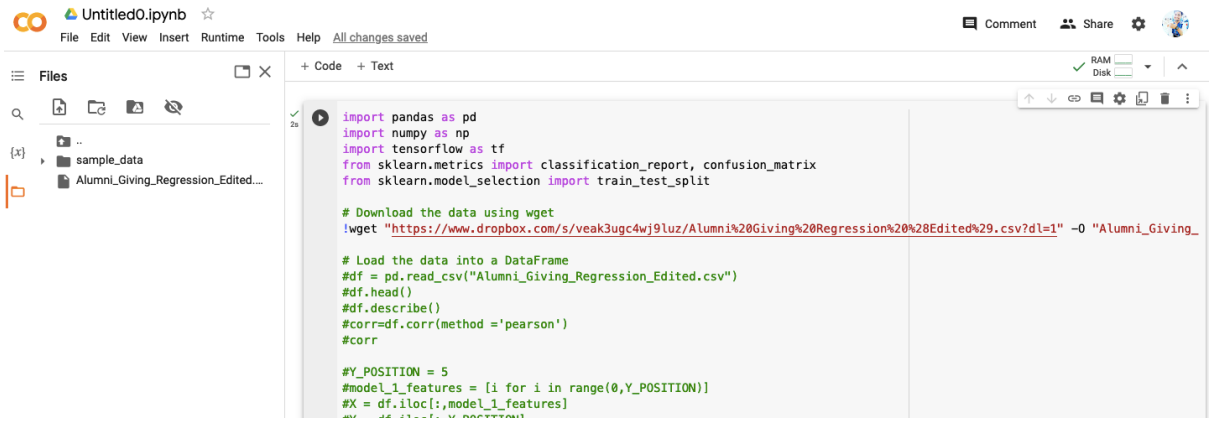
In-Lab

Task 1

```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split

# Download the data using wget
!wget
"https://www.dropbox.com/s/veak3ugc4wj9luz/Alumni%20Giving%20Regressi
on%20%28Edited%29.csv?dl=1" -O "Alumni_Giving_Regression_Edited.csv"
```

Output:

We have successfully downloaded the dataset from link as you can see it in figure.



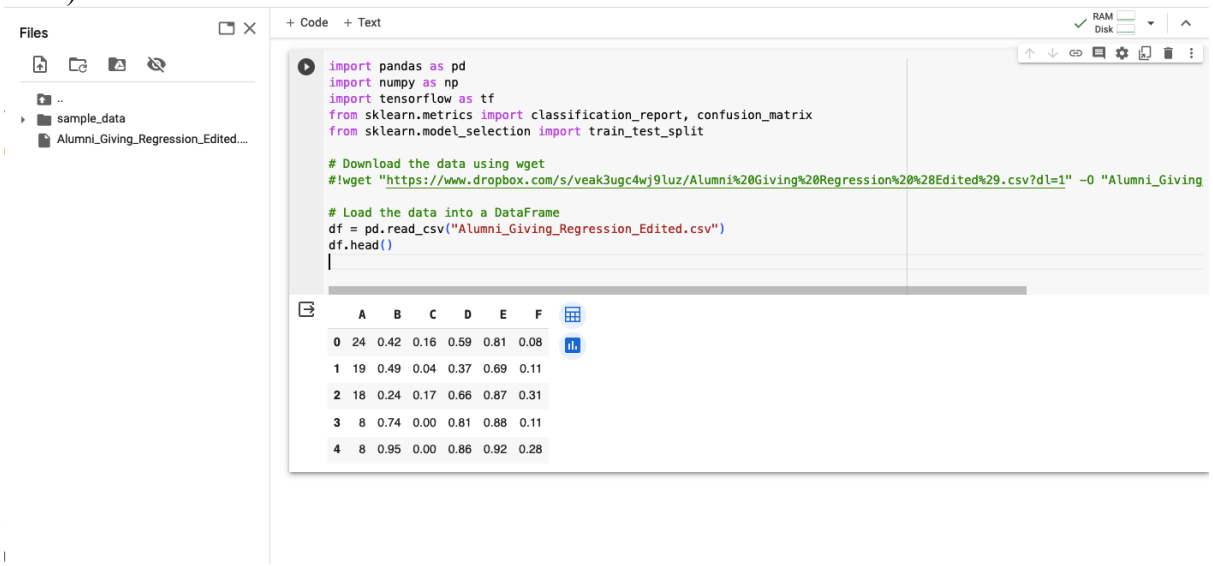
Task 2

```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split

# Download the data using wget
#!wget
"https://www.dropbox.com/s/veak3ugc4wj9luz/Alumni%20Giving%20Regressi
on%20%28Edited%29.csv?dl=1" -O "Alumni_Giving_Regression_Edited.csv"

# Load the data into a DataFrame
df = pd.read_csv("Alumni_Giving_Regression_Edited.csv")
```

```
df.head()
```

Output:

We are Loading and displaying the dataset using pandas. The pd.read_csv("Alumni_Giving_Regression_Edited.csv") function reads the dataset into a DataFrame, and df.head() displays the first few rows of the DataFrame (By Default show 5 rows).

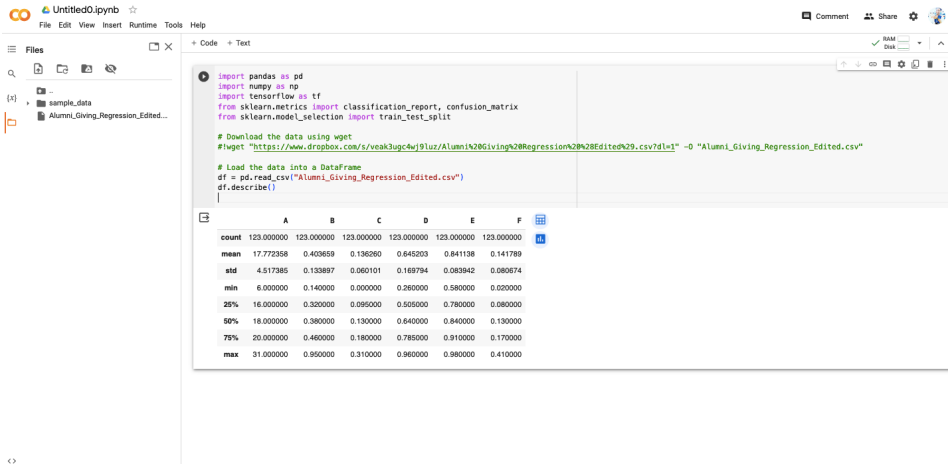


Task 3

```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split

# Download the data using wget
#!wget
"https://www.dropbox.com/s/veak3ugc4wj9luz/Alumni%20Giving%20Regressi
on%20%28Edited%29.csv?dl=1" -O "Alumni_Giving_Regression_Edited.csv"

# Load the data into a DataFrame
df = pd.read_csv("Alumni_Giving_Regression_Edited.csv")
df.describe()
```

Output:

We are summarizing the dataset statistics. The df.describe() function provides summary statistics (count, mean, std, min, max, etc.) for each numerical column in the DataFrame.

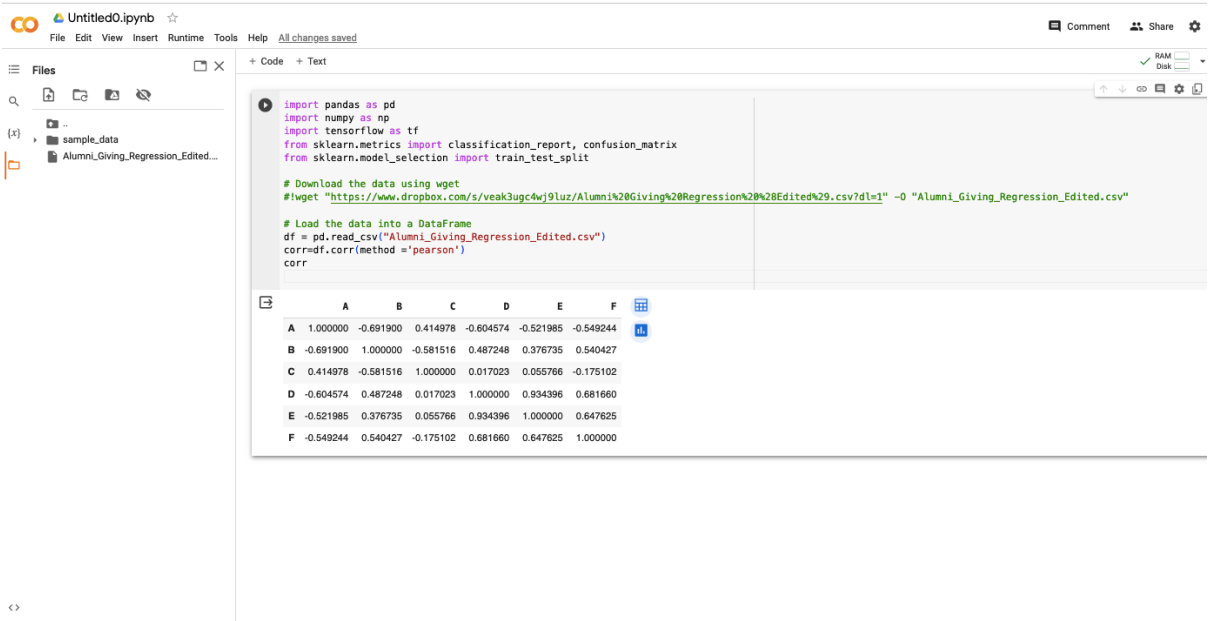


Task 4

```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split

# Download the data using wget
#!wget
"https://www.dropbox.com/s/veak3ugc4wj9luz/Alumni%20Giving%20Regressi
on%20%28Edited%29.csv?dl=1" -O "Alumni_Giving_Regression_Edited.csv"

# Load the data into a DataFrame
df = pd.read_csv("Alumni_Giving_Regression_Edited.csv")
corr=df.corr(method ='pearson')
corr
```

Output:

We are calculating and displaying the Pearson correlation matrix for the dataset. The corr variable stores the correlation matrix calculated using df.corr(method='pearson').



Task 5

```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split

# Download the data using wget
#!wget
"https://www.dropbox.com/s/veak3ugc4wj9luz/Alumni%20Giving%20Regressi
on%20%28Edited%29.csv?dl=1" -O "Alumni_Giving_Regression_Edited.csv"

# Load the data into a DataFrame
df = pd.read_csv("Alumni_Giving_Regression_Edited.csv")
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)
```

Output:

We are splitting the dataset into training and testing sets for machine learning. The code uses train_test_split from scikit-learn to split the features (X) and the target variable (Y) into training and testing sets.

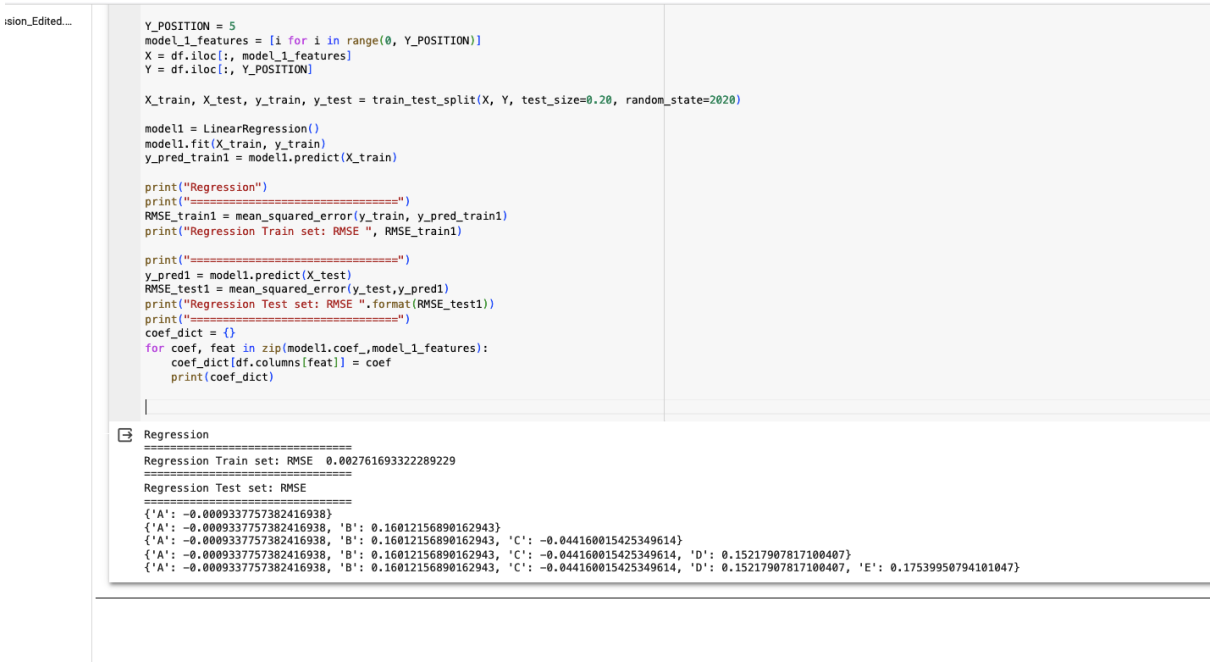
Task 6

```
import pandas as pd
import numpy as np
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Load the data into a DataFrame
df = pd.read_csv("Alumni_Giving_Regression_Edited.csv")

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)

model1 = 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 ", 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)
```

Output:

We are building a linear regression model, fitting it to the training data, and evaluating the model's performance. The code uses scikit-learn to create a linear regression model, train it,

make predictions on the training and testing data, and calculate the Root Mean Squared Error (RMSE) for model evaluation. It also displays the coefficients of the model for each feature.



```
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)

model1 = 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 ", 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
================================
{'A': -0.0009337757382416938}
{'A': -0.0009337757382416938, 'B': 0.16012156890162943}
{'A': -0.0009337757382416938, 'B': 0.16012156890162943, 'C': -0.044160015425349614}
{'A': -0.0009337757382416938, 'B': 0.16012156890162943, 'C': -0.044160015425349614, 'D': 0.15217907817100407}
{'A': -0.0009337757382416938, 'B': 0.16012156890162943, 'C': -0.044160015425349614, 'D': 0.15217907817100407, 'E': 0.17539950794101047}
```

Note:

The tasks are performed using common libraries like pandas, numpy, scikit-learn (sklearn), and TensorFlow. However, we have flexibility in choosing alternative libraries or tools for similar tasks like Dask, Vaex, cuDF (for large datasets or distributed computing).We can use Matplotlib or Seaborn for more advanced data visualization. We can use NumPy for basic statistics, and Pandas Profiling for more detailed data profiling. We can use SciPy for correlation functions.We can use TensorFlow's built-in data preprocessing functions.We can use other machine learning libraries like XGBoost, LightGBM, or scikit-learn for regression models.