Lab Report

Pre-Lab:

```
!wget -O "./Alumni Giving Regression (Edited).csv"
"https://www.dropbox.com/s/veak3ugc4wj9luz/Alumni%20Giving%20Regression%20%28
Edited%29.csv"
```

```
!ls
```

Code output:

```
'Alumni Giving Regression (Edited).csv'   -quiet   sample_data
```

In-Lab Task 1:

```
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
import matplotlib.pyplot as plt
```

In lab Task 2:

```
np.random.seed(7)
data = pd.read_csv('Alumni Giving Regression (Edited).csv')
dd_df_1 = data.head()
```

In lab Task 3:

```
data.head()
```

Code output:

	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

```
data.describe()
```

Code output:

	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

In lab Task 4:

```
corr=data.corr(method ='pearson')
print(corr)
```

Code output:

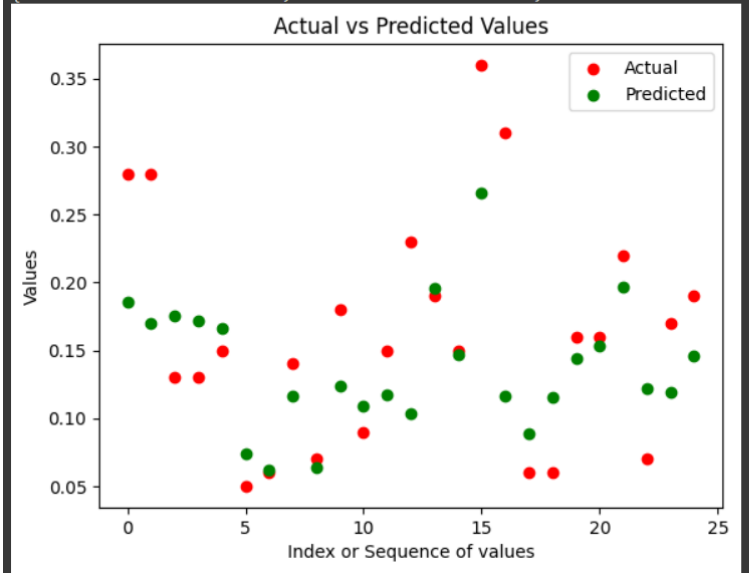
```
          A         B         C         D         E         F
A  1.000000 -0.691900  0.414978 -0.604574 -0.521985 -0.549244
B -0.691900  1.000000 -0.581516  0.487248  0.376735  0.540427
C  0.414978 -0.581516  1.000000  0.017023  0.055766 -0.175102
D -0.604574  0.487248  0.017023  1.000000  0.934396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

In lab Task 5:

```
[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)
```

```python
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)

x_values = np.arange(len(y_test))
plt.scatter(x_values, y_test, color = 'red', label = 'Actual')
plt.scatter(x_values, y_pred1, color = 'green', label = 'Predicted')
plt.xlabel('Index or Sequence of values')
plt.ylabel('Values')
plt.title('Actual vs Predicted Values')
plt.legend()
plt.show()
```

```
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}
```



Post-Lab

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# Load the dataset (Titanic dataset used as an example)
url =
'https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv
'
data = pd.read_csv(url)

# Select relevant features and target variable
features = ['Pclass', 'Age', 'SibSp', 'Parch', 'Fare']
target = 'Survived'

# Handle missing values (e.g., impute Age with median)
data['Age'].fillna(data['Age'].median(), inplace=True)

# Create X (features) and y (target)
X = data[features]
y = data[target]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Feature scaling/normalization
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Initialize and fit a linear regression model
model = LinearRegression()
model.fit(X_train_scaled, y_train)

# Predict on the test set
y_pred = model.predict(X_test_scaled)

# Calculate RMSE
RMSE = mean_squared_error(y_test, y_pred, squared=False)
print(f"Linear Regression RMSE: {RMSE}")

# Plotting actual vs predicted values in red (actual) and green (predicted)
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_test, color='red', label='Actual')  # Plotting actual
values in red
plt.scatter(y_test, y_pred, color='green', label='Predicted')  # Plotting
predicted values in green
```

```
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted Values (Linear Regression)')
plt.legend()
plt.show()
```

Code output:

```
Linear Regression RMSE: 0.4323515379937008
```