Irfan Ahmed

ICP 10

Due Date: 11/3/2020

a. What you learned in the ICP

For this ICP the goal was to design a Deep Q learning Network (DNQ) using Keras & OpenAI Gym, for cartpole game. The basic idea behind reinforcement learning is that it receives feedback in the form of rewards. The agent's utility is defined by the reward function and learn to maximize the expected rewards. The learning is based on the outcomes from the observed samples. The Cartpole environment is composed of observations, reward, and action. The observation consists of an array containing 4 float values. It has the cart velocity, angular position, and velocity of the pole. The reward is scalar float values consisting of 0 and 1's. The 0 means move to the left and 1 means move to the right. This is just basic outline what I learned in the process while working on the ICP.

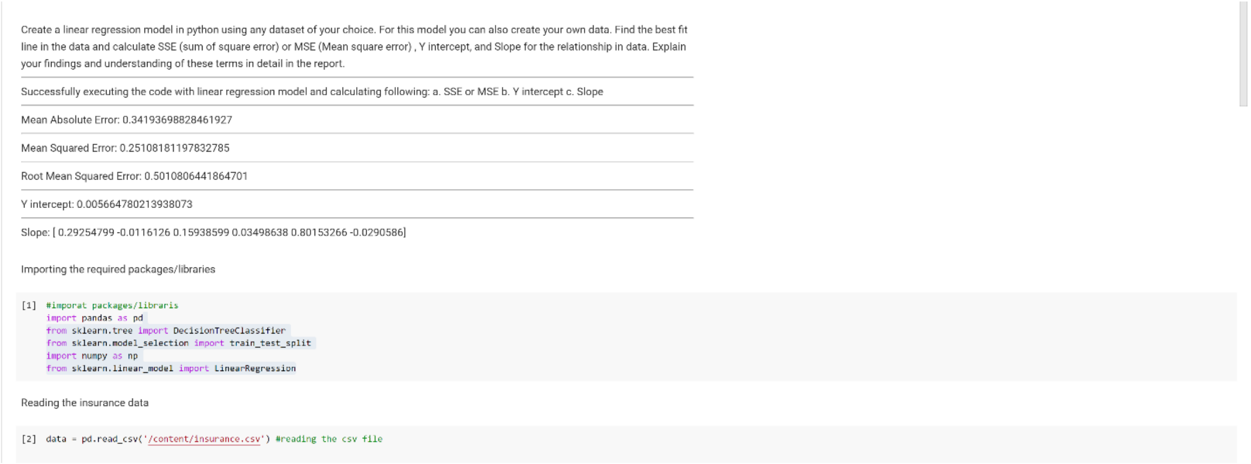
b. ICP description what was the task you were performing

- Imported the required libraries
- created environment using gym.make for Cartpole-v0. Looked at the observation state amd action state
- Actual environment setup and rendered the results
- Setup the parameters like episodes, explorations rate, memory size, learning size, memory, etc..
- Build model using keras. Model = Sequential ()
- Created functions for memory, exploration rate, preprocess to determine the format is in the right format, relay function, and run function.

c. Challenges that you faced

This ICP was more difficult than I thought it would be. There is lot of information on the internet, but it was very difficult to determine where I should begin. I had difficulty in putting the model and training together and figuring out how to display the visualization on my screen. Encountered lot of confusion why certain parameters were used, and how to train the data before applying it to my model. At the end add the most difficult time running my model on the screen.

d. Screen shots that shows the successful execution of each required step of your code



Create a linear regression model in python using any dataset of your choice. For this model you can also create your own data. Find the best fit line in the data and calculate SSE (sum of square error) or MSE (Mean square error) , Y intercept, and Slope for the relationship in data. Explain your findings and understanding of these terms in detail in the report.

Successfully executing the code with linear regression model and calculating following: a. SSE or MSE b. Y intercept c. Slope

Mean Absolute Error: 0.34193698828461927

Mean Squared Error: 0.25108181197832785

Root Mean Squared Error: 0.5010806441864701

Y intercept: 0.005664780213938073

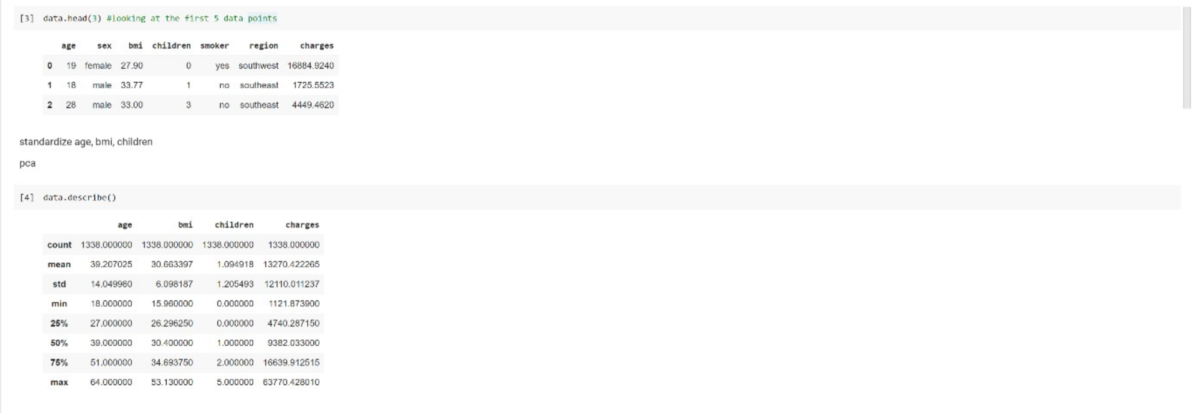
Slope: [0.29254799 -0.0116126 0.15938599 0.03498638 0.80153266 -0.0290586]

Importing the required packages/libraries

```
[1]  #imporat packages/libraris
     import pandas as pd
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.model_selection import train_test_split
     import numpy as np
     from sklearn.linear_model import LinearRegression
```

Reading the insurance data

```
[2]  data = pd.read_csv('/content/insurance.csv') #reading the csv file
```



```
[3]  data.head(3) #looking at the first 5 data points
```

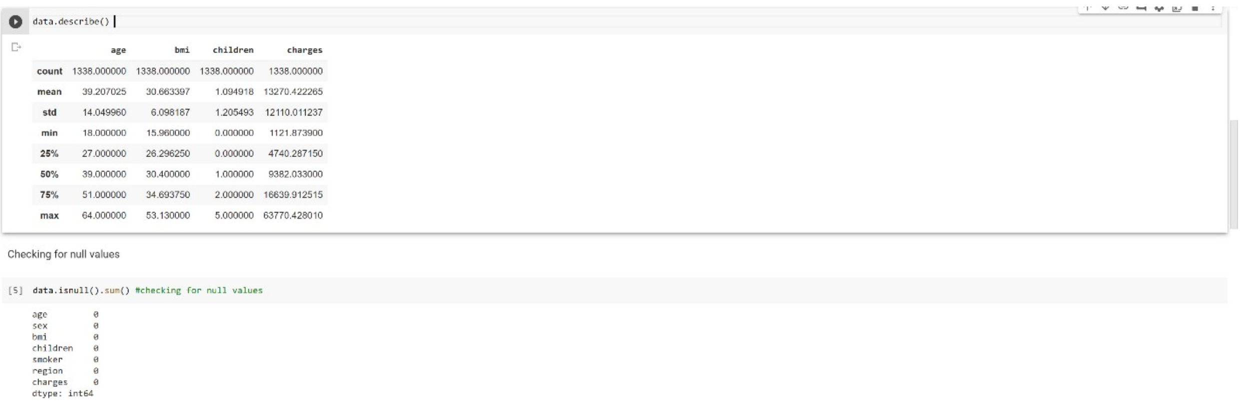
	age	sex	bmi	children	smoker	region	charges
0	19	female	27.90	0	yes	southwest	16884.9240
1	18	male	33.77	1	no	southeast	1725.5523
2	28	male	33.00	3	no	southeast	4449.4620

standardize age, bmi, children

pca

```
[4]  data.describe()
```

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010



```
data.describe()
```

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

Checking for null values

```
[5]  data.isnull().sum() #checking for null values

     age         0
     sex         0
     bmi         0
     children    0
     smoker      0
     region      0
     charges     0
     dtype: int64
```

Will need to convert to number format for sex, region and smoker. 0 is smoker and 1 is not a smoker. 0 is female and 1 is male. value for region is based on the location.

```
[6]  from sklearn.preprocessing import LabelEncoder
     le = LabelEncoder()
     le.fit(data.sex.drop_duplicates())
     data.sex = le.transform(data.sex) #sex male or female

     #print(data.sex)

     le.fit(data.smoker.drop_duplicates())
     data.smoker = le.transform(data.smoker) #smoker yes or no

     #print(data.smoker)

     le.fit(data.region.drop_duplicates()) #region
     data.region = le.transform(data.region)

     #print(data.region)
```

```
     data.head(5)
```

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	1	3	16884.92400
1	18	1	33.770	1	0	2	1725.55230
2	28	1	33.000	3	0	2	4449.46200
3	33	1	22.705	0	0	1	21984.47061
4	32	1	28.880	0	0	1	3866.85520

```
[8]  from sklearn.decomposition import PCA

     pca = PCA(whiten=True)
     pca.fit(data)
     variance = pd.DataFrame(pca.explained_variance_ratio_)
     np.cumsum(pca.explained_variance_ratio_)

     array([0.99999851, 0.99999974, 0.99999998, 0.99999999, 1.        ,
            1.        , 1.        ])
```

Creating a features and X and y variables

```
[9]  feature_cols = ['age', 'sex', 'bmi', 'children', 'smoker', 'region']

     X = data[feature_cols]
     y = data.charges
```

```
[11] #standardized X abd y
     X = preprocessing.scale(X)
     y = preprocessing.scale(y)
```

Training the X and y datasets

```
[12] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 1)
```

Using the linear regression model for analysis

```
[13] linear_model = LinearRegression()
```

```
[14] model_fit = linear_model.fit(X_train,y_train) #fitting the training and test data
```

```
[15] linear_model.score(X_train,y_train) #looking at the score

     0.7544083642384213
```

```
[16] pred = linear_model.predict(X_test) #predction value
```

```
[17] from sklearn.metrics import r2_score
     r2_score(y_test, pred)#R squared

     0.740367716897532
```

For retrieving the slope (coefficient of x)

```
[18] linear_model.coef_ #gives you an array of weights estimated by linear regression. It is of shape (ntargets, nfeatures)

     array([ 0.29254799, -0.0116126 ,  0.15938599,  0.03498638,  0.80153266,
            -0.0290586 ])
```

To retrieve the intercept

```
[19] linear_model.intercept_

     0.005664780213938073
```

```
[20] from sklearn.metrics import mean_squared_error
     from sklearn import metrics
```

```
[22] from sklearn import metrics
     print('Mean Absolute Error: ', metrics.mean_absolute_error(y_test, pred))
     print('Mean Squared Error: ', metrics.mean_squared_error(y_test, pred))
     print('Root Mean Squared Error: ', np.sqrt(metrics.mean_squared_error(y_test, pred)))
     print('Y intercept: ', linear_model.intercept_)
     print('Slope: ', linear_model.coef_)

     Mean Absolute Error:  0.34193698828461927
     Mean Squared Error:  0.25108181197832785
     Root Mean Squared Error:  0.5010806441864701
     Y intercept:  0.005664780213938073
     Slope:  [ 0.29254799 -0.0116126   0.15938599  0.03498638  0.80153266 -0.0290586 ]
```

e. Output file link if applicable

https://github.com/UMKC-APL-BigDataAnalytics/icp10-irfancheemaa

f. Video link (YouTube or any other publicly available video platform)

https://umkc.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=6368a223-639e-4770-ad23-ac69000e5bf9

g. Any inside about the data or the ICP in general

I enjoyed the concept, but it was a bit difficult to run most of the stuff in google colab. I think a small outline would have been nice to follow and get an idea how the structure the code.