Computer Assignment #0 – Introduction to Artificial Intelligence

Ghazal Kalhor

810196675

kalhorghazal1378@gmail.com

Abstract **— In this computer assignment, we want to learn how to work with some of the most important libraries in Python such as NumPy, Pandas, and matplotlib that will help us in learning Artificial Intelligence; this is done by working with data frames and using them to predict a simple linear regression model.**

Keywords **— Python, NumPy, Pandas, Matplotlib, Artificial Intelligence**

I. Introduction

The goal of this computer assignment is to design a simple model for prediction of the probability of acceptance of graduate students in Universities in the USA in master degree. This model has some inputs such as GRE Score, TOEFL Score, and CGPA that we will use them in prediction.

II. Downloading and Exploring Data Frames

We are given the information of about 400 graduate students in a .csv file.

A. Reading File and Calling Functions

In this part we read context of the given .csv file by using the function "pd.read_csv" from Pandas and saved it in a data frame. Then we called some of the functions in this library, such as "head", "describe", and "info" on the data frame that mentioned above. The results are shown in Fig. 1, 2, 3.

Code 1
Reading File and Calling Functions

```
df = pd.read_csv('AdmissionPredict.csv')
print(df.head())

print(df.describe())

print(df.info())
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337.0	118.0	4	4.5	4.5	9.65	1	0.92
1	2	324.0	107.0	4	4.0	4.5	8.87	1	NaN
2	3	316.0	NaN	3	3.0	3.5	8.00	1	0.72
3	4	NaN	110.0	3	3.5	2.5	8.67	1	0.80
4	5	314.0	103.0	2	2.0	3.0	8.21	0	0.65

Fig. 1 – Output of "head" Function

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	400.000000	378.000000	380.000000	400.000000	400.000000	400.000000	380.000000	400.000000	384.000000
mean	200.500000	316.759259	107.386842	3.087500	3.400000	3.452500	8.684737	0.547500	0.724375
std	115.614301	11.415599	6.048645	1.143728	1.006869	0.898478	0.599167	0.498362	0.142964
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	0.000000	0.340000
25%	100.750000	308.250000	103.000000	2.000000	2.500000	3.000000	8.170000	0.000000	0.640000
50%	200.500000	317.000000	107.000000	3.000000	3.500000	3.500000	8.640000	1.000000	0.730000
75%	300.250000	325.000000	112.000000	4.000000	4.000000	4.000000	9.080000	1.000000	0.830000
max	400.000000	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	1.000000	0.970000

Fig. 2 – Output of "describe" Function

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 9 columns):
Serial No.           400 non-null int64
GRE Score            378 non-null float64
TOEFL Score          380 non-null float64
University Rating    400 non-null int64
SOP                  400 non-null float64
LOR                  400 non-null float64
CGPA                 380 non-null float64
Research             400 non-null int64
Chance of Admit      384 non-null float64
dtypes: float64(6), int64(3)
memory usage: 28.2 KB
None
```

Fig. 3 – Output of "info" Function

B. Counting Missing Values

In this This part we used some of the functions of Pandas in order to calculate the number of missing values in each column of the data frame. The result is shown in Table 1.

Code 2
Counting Missing Values

```
print(df.isna().sum())
```

```
Serial No.            0
GRE Score            22
TOEFL Score          20
University Rating     0
SOP                   0
LOR                   0
CGPA                 20
Research              0
Chance of Admit      16
dtype: int64
```

Fig. 4 – Output of Code 2

TABLE 1
NUMBER OF MISSING VALUES

Column	Number of Missing Values
Serial No.	0
GRE Score	22
TOEFL Score	20
University Rating	0
SOP	0
LOR	0
CGPA	20
Research	0
Chance of Admit	16

C. Filling Missing Values

In this part we filled each missing cell, with the average of the corresponding column to that cell; this is done by composition of functions "df.mean" and "df.fillna". We know that the column "Chance of Admit" is our target variable, so we shouldn't replace the missing cells of that. The result is shown in Fig. 5.

CODE 3
FILLING MISSING VALUES

```
df['GRE Score'] = df['GRE Score'].fillna(df['GRE Score'].mean())
df['TOEFL Score'] = df['TOEFL Score'].fillna(df['TOEFL Score'].mean())
df['University Rating'] = df['University Rating'].fillna(df['University Rating'].mean())
df['SOP'] = df['SOP'].fillna(df['SOP'].mean())
df['LOR '] = df['LOR '].fillna(df['LOR '].mean())
df['CGPA'] = df['CGPA'].fillna(df['CGPA'].mean())
df['Research'] = df['Research'].fillna(df['Research'].mean())
print(df.isna().sum())
```

```
Serial No.           0
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit     16
dtype: int64
```

Fig. 5 – Output of Code 3

III. CORRELATION AND RELATIONSHIP BETWEEN CHARACTERISTICS AND TARGET VARIABLE

We want to predict the chance of admit for each student based on the input characteristics, so in this section we will show these relations on graphs with different labels.

A. Drawing Plots

In this part we drew a plot for each characteristic to show its relation with "Chance of Admit"; this is done by using "scatterplot" function from Matplotlib. The results are shown in Fig. 6, 7, 8, 9, 10, 11, 12, 13.

CODE 4
DRAWING PLOTS

```
plt.scatter('Serial No.', 'Chance of Admit', c='#b5347e', data=df)
plt.xlabel('Serial No.')
plt.ylabel('Chance of Admit')
plt.show()

plt.scatter('GRE Score', 'Chance of Admit', c='#9a34b5', data=df)
plt.xlabel('GRE Score')
plt.ylabel('Chance of Admit')
plt.show()

plt.scatter('TOEFL Score', 'Chance of Admit', c='#1cc32f', data=df)
plt.xlabel('TOEFL Score')
plt.ylabel('Chance of Admit')
plt.show()

plt.scatter('University Rating', 'Chance of Admit', c='#1691a7', data=df)
plt.xlabel('University Rating')
plt.ylabel('Chance of Admit')
plt.show()

plt.scatter('SOP', 'Chance of Admit', c='#de6722', data=df)
plt.xlabel('SOP')
plt.ylabel('Chance of Admit')
plt.show()

plt.scatter('LOR ', 'Chance of Admit', c='#e5d700', data=df)
plt.xlabel('LOR ')
plt.ylabel('Chance of Admit')
plt.show()

plt.figure()
plt.scatter('CGPA', 'Chance of Admit', c='#e500a9', data=df)
plt.xlabel('CGPA')
plt.ylabel('Chance of Admit')
plt.show()

plt.scatter('Research', 'Chance of Admit', c='#eb8def', data=df)
plt.xlabel('Research')
plt.ylabel('Chance of Admit')
plt.show()
```

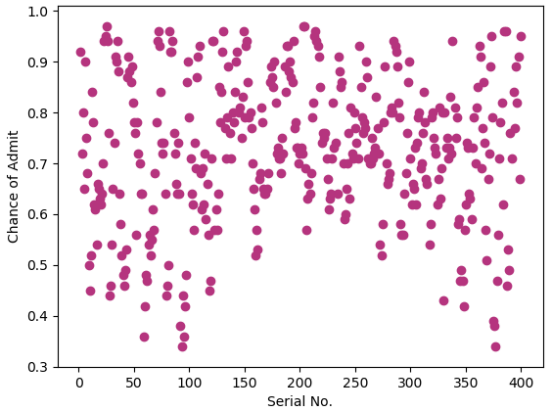

Fig. 6 – "Serial No." versus "Chance of Admit"

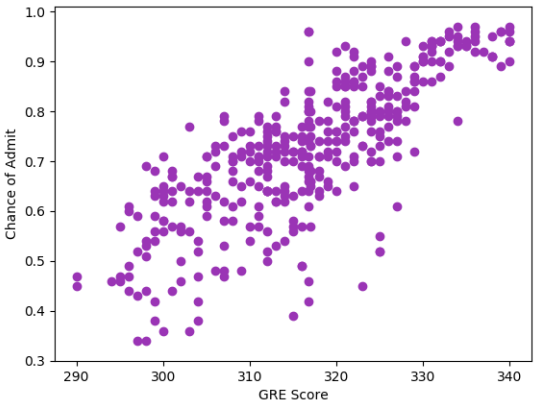

Fig. 7 – "GRE Score" versus "Chance of Admit"

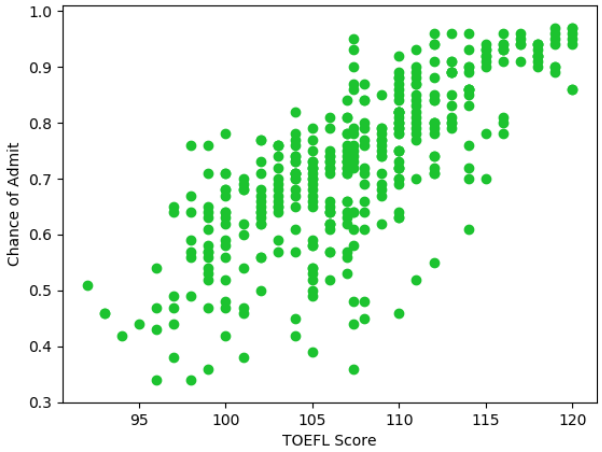

Fig. 8 – "TOEFL Score" versus "Chance of Admit"

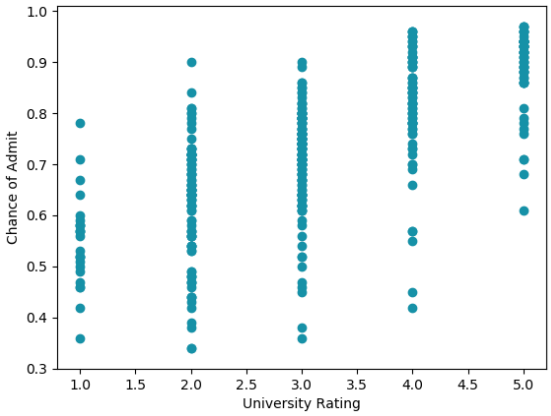

Fig. 9 – "University Rating" versus "Chance of Admit"

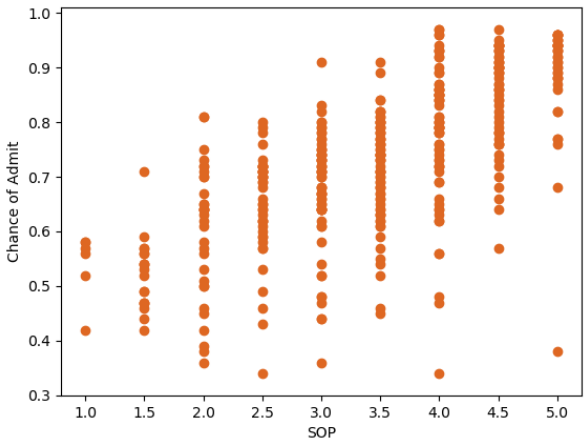

Fig. 10 – "SOP" versus "Chance of Admit"

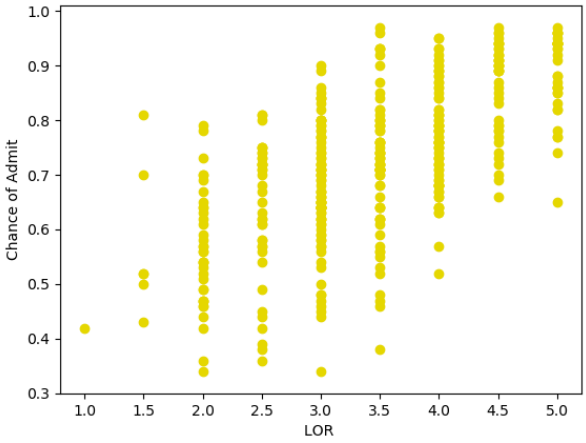

Fig. 11 – "LOR" versus "Chance of Admit"

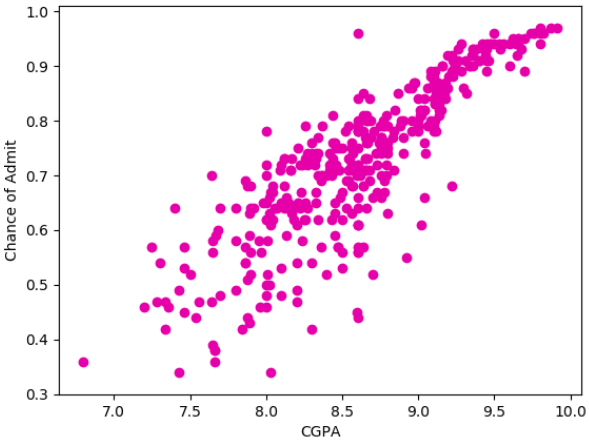

Fig. 12 – "CGPA" versus "Chance of Admit"

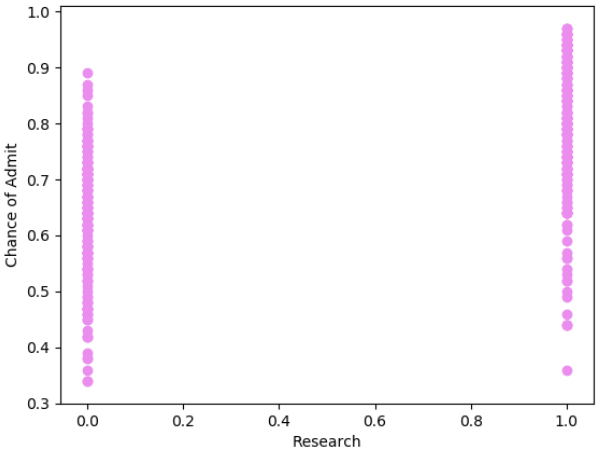

Fig. 13 – "Research" versus "Chance of Admit"

B. The Most Linear Characteristic

In last part we drew different graphs. If we compare them, we conclude that the most linear characteristic is "CGPA". Although we know that there are more than one linear characteristic; we will choose "CGPA". This is rational, because "CGPA" is one of the most criteria in evaluating performance of a student.

IV. Working with Data Frames

In this section, we want do some tasks in a vectorized way; this is done by using some functions from Pandas and NumPy.

A. Filtering Students

In this part we filtered students with "CGPA" at least 9 and "TOEFL Score" at least 110, with assumption that there is a specific university that has above conditions for accepting students. Furthermore we obtained number of students with the above conditions. The result is shown in Fig. 14.

Code 5
Filtering Students

```
CGPA_filtered = df[df['CGPA'] >= 9]
TOEFL_filtered = CGPA_filtered[CGPA_filtered['TOEFL Score'] >= 110]
array = np.array(TOEFL_filtered['Serial No.'])
print("Number of Students with CGPA >= 9 and TOEFL Score >= 110 :")
print(array.size)
```

```
ghazal@ghazal-Surface-Pro-4:~/Desktop/AI_CA0$ python3 ca0.py
Number of Students with CGPA >= 9 and TOEFL Score >= 110 :
97
```

Fig. 14 – Output of Code 5

B. Average of a Characteristic

In this part we filtered graduate students with "University Rating" in range of 1 to 5 separately, then we calculated the average of "GRE Scores" for these students. The results are shown in Table 2.

Code 6
Average of a Characteristic

```
rank1 = df[df['University Rating'] == 1]
print("Average of GRE Scores of Students from Universities with rank 1 :")
print(rank1['GRE Score'].mean())

rank2 = df[df['University Rating'] == 2]
print("Average of GRE Scores of Students from Universities with rank 2 :")
print(rank2['GRE Score'].mean())

rank3 = df[df['University Rating'] == 3]
print("Average of GRE Scores of Students from Universities with rank 3 :")
print(rank3['GRE Score'].mean())

rank4 = df[df['University Rating'] == 4]
print("Average of GRE Scores of Students from Universities with rank 4 :")
print(rank4['GRE Score'].mean())

rank5 = df[df['University Rating'] == 5]
print("Average of GRE Scores of Students from Universities with rank 5 :")
print(rank5['GRE Score'].mean())
```

```
Average of GRE Scores of Students from Universities with rank 1 :
303.15384615384613
Average of GRE Scores of Students from Universities with rank 2 :
309.7528556593977
Average of GRE Scores of Students from Universities with rank 3 :
315.9346978557505
Average of GRE Scores of Students from Universities with rank 4 :
324.07507507507506
Average of GRE Scores of Students from Universities with rank 5 :
327.9546296296296
```

Fig. 15– Output of Code 6

Table 2
Average of a Characteristic

University Rating	Average of GRE Scores
1	303.15384615384613
2	309.7528556593977
3	315.9346978557505
4	324.07507507507506
5	327.9546296296296

V. One-Variable Linear Regression

In this section, we want to design a linear estimator based on the selected characteristic in last section ("CGPA") in order to estimate the chance of admit. In other words, we want to estimate the line that coincides the graph of "CGPA" versus "Chance of Admit"; this is done by using the following formulas.

$$h(x) = \theta_0 + \theta_1 x$$

$$SS_{xy} = \sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) = \sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}$$

$$SS_{xx} = \sum_{i=1}^{n}(x_i - \bar{x})^2 = \sum_{i=1}^{n} {x_i}^2 - n(\bar{x})^2$$

$$\theta_1 = \frac{SS_{xy}}{SS_{xx}}$$

$$\theta_0 = \bar{y} - \theta_1 \bar{x}$$

A. *Making a New Data Frame*

In this part we made a new data frame from the data of selected characteristic in last section (“CGPA”) and target column (“Chance of Admit”). (Code 7)

CODE 7
MAKING A NEW DATASET

```
new_df = df.loc[:, ['CGPA', 'Chance of Admit']]
x = df['CGPA']
y = df['Chance of Admit']
xm = df['CGPA'].mean()
ym = df['Chance of Admit'].mean()
n = x.size
```

B. *Minimizing Cost Function*

In this part we want to find coefficients that mentioned above, in order to minimize the cost function with the following formula. For this goal we used the formulas that we mentioned in last part. The result is shown in Fig. 16, 17.

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

CODE 8
MINIMIZING COST FUNCTION

```
SSxy = x * y - n * xm * ym
SSxy = SSxy.mean() * n

SSxx = x * x - n * xm * xm
SSxx = SSxx.mean() * n

theta1 = SSxy / SSxx
theta0 = ym - theta1 * xm

h = theta0 + theta1 * x

j = (h - y)**2
j = j.mean() / 2

print("Mean Square Error :")
print(j)

plt.scatter('CGPA', 'Chance of Admit', c='#e500a9', data=df)
plt.scatter(x, h, c='g')
plt.xlabel('CGPA')
plt.ylabel('Chance of Admit')
plt.show()
```

```
Mean Square Error :
0.005402975441154513
```

Fig. 16– Mean Square Error

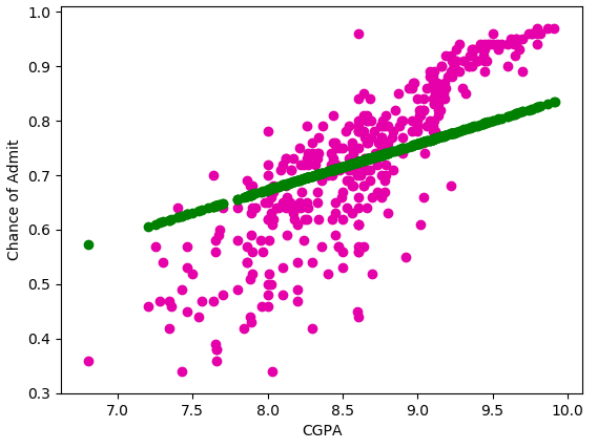

Fig. 17– Estimated line

C. *Estimating Missing Chance of Admit*

In this part we estimated the missing chance of admit, based on the equation of the line that we obtained in last part. The output of our model is shown in Table 3.

CODE 9
ESTIMATING MISSING CHANCE OF ADMIT

```
NaN_filtered = df[pd.isna(df['Chance of Admit'])]
print("Students with Missing Chance of Admits :")
print(NaN_filtered)

NaN_filtered['Chance of Admit'] = theta0 + theta1 * NaN_filtered['CGPA']
print("Output of my Model to Fill Missing Chance of Admits :")
print(NaN_filtered)
```

Students with Missing Chance of Admits :

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
1	2	324.000000	107.000000	4	4.0	4.5	8.87	1	NaN
57	58	298.000000	99.000000	2	4.0	2.0	7.60	0	NaN
135	136	314.000000	109.000000	4	3.5	4.0	8.77	1	NaN
143	144	340.000000	120.000000	4	4.5	4.0	9.92	1	NaN
171	172	316.759259	117.000000	5	4.0	4.5	9.07	1	NaN
218	219	324.000000	110.000000	4	3.0	3.5	8.97	1	NaN
232	233	312.000000	107.386842	2	2.5	3.5	8.27	0	NaN
251	252	316.000000	99.000000	2	2.5	3.0	9.00	0	NaN
272	273	294.000000	95.000000	1	1.5	1.5	7.64	0	NaN
294	295	316.759259	101.000000	2	2.5	2.0	8.32	1	NaN
315	316	308.000000	104.000000	2	2.5	3.0	8.07	0	NaN
338	339	323.000000	108.000000	5	4.0	4.0	8.74	1	NaN
357	358	301.000000	104.000000	2	3.5	3.5	7.89	1	NaN
381	382	319.000000	105.000000	3	3.0	3.5	8.67	1	NaN
390	391	314.000000	102.000000	2	2.0	2.5	8.24	0	NaN
396	397	325.000000	107.000000	3	3.0	3.5	9.11	1	NaN

Fig. 18– Students with Missing Chance of Admit

Output of my Model to Fill Missing Chance of Admits :

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
1	2	324.000000	107.000000	4	4.0	4.5	8.87	1	0.746705
57	58	298.000000	99.000000	2	4.0	2.0	7.60	0	0.639794
135	136	314.000000	109.000000	4	3.5	4.0	8.77	1	0.738287
143	144	340.000000	120.000000	4	4.5	4.0	9.92	1	0.835096
171	172	316.759259	117.000000	5	4.0	4.5	9.07	1	0.763542
218	219	324.000000	110.000000	4	3.0	3.5	8.97	1	0.755124
232	233	312.000000	107.386842	2	2.5	3.5	8.27	0	0.696196
251	252	316.000000	99.000000	2	2.5	3.0	9.00	0	0.757649
272	273	294.000000	95.000000	1	1.5	1.5	7.64	0	0.643162
294	295	316.759259	101.000000	2	2.5	2.0	8.32	1	0.700405
315	316	308.000000	104.000000	2	2.5	3.0	8.07	0	0.679360
338	339	323.000000	108.000000	5	4.0	4.0	8.74	1	0.735762
357	358	301.000000	104.000000	2	3.5	3.5	7.89	1	0.664207
381	382	319.000000	105.000000	3	3.0	3.5	8.67	1	0.729869
390	391	314.000000	102.000000	2	2.0	2.5	8.24	0	0.693671
396	397	325.000000	107.000000	3	3.0	3.5	9.11	1	0.766909

Fig. 19– Estimated Chance of Admits

TABLE 3
ESTIMATING MISSING CHANCE OF ADMIT

Serial No.	Estimated Chance of Admit
1	0.746705
57	0.639794
135	0.738287
143	0.835096
171	0.763542
218	0.755124
232	0.696196
251	0.757649
272	0.643162
294	0.700405
315	0.679360
338	0.735762
357	0.664207
381	0.729869
390	0.693671
396	0.766909

VI. CONCLUSIONS

In this computer assignment we learned that Python libraries are very powerful tools in Artificial Intelligence. Also we were introduced one of the applications of Artificial Intelligence in real life.

REFERENCES

[1] COMPUTER ASSIGNMENT MANUAL, Computer Assignment 0, *Introduction to Artificial Intelligence*

[2] COPUTER ARCITECTURE FALL 98, COMPUTER ASSIGNMENT MANUAL, Computer Assignment 1, *Linear Regression*