**CSE 574 - MACHINE LEARNING**

**PROJECT REPORT FOR HOMEWORK 1**

* Sujit Singh (UBITName: sujitjit, Person #: 50247206)
* Gautam Shende (UBITName: gautamav, Person #: 50245840)

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **SR. NO.** | **Particulars** |
| 1 | Summary |
| 2 | Implementation |
| 3 | Results |
| 4 | Observation & Conclusion |
| 5 | References |

**1. Summary**

This is a report that covers an assignment on basic statistics and probability concepts involved in Machine learning that lead us towards the use of Bayesian Networks. Some of the concepts covered in this assignment are mean, variance, standard deviation, covariance, correlation and loglikelihood. The assignment was completed in **Python 2.7** taking help of various libraries such as numpy[1] and scipy[3] for statistical operations, pandas[2] for data handling and matplotlib[4] for plotting graphs and interpreting through visualization.

**2. Implementation**

The submission is divided into two files:

1. Main.py
2. Exec.py

The **Main.py** is used for printing the initial submission details like UBitName and personNumber.

It then simply calls the exec.py using execfile() function in python.

This file is the entry point to the program.

**Exec.py** is used for doing all the computations and all the results are obtained using this python file. There are 7 functions that are used for performing core computations of the problem.

1. readExcelFile(fileName, sheetName)
2. allocateData(dataFrame,columnNames)
3. calculate(dataFrame,columnNames)
4. calculateCovariance(dataFrame, columnNames)
5. calculateCorrelationCoeff(dataFrame, columnNames)
6. calculateMultivariateLikelihood(dataFrame, columnNames)
7. calculateMultivariateLikelihood(dataFrame, columnNames)

Here is what each of the functions do:

1. ***readExcelFile(fileName, sheetName):***

This function is used to read data from an excel file using the read\_excel function in pandas[2] and construct a dataframe from it.

Input parameters:

1. fileName: This is the name of the excel file from which data is to be read
2. sheetName: This is the name of the sheet to be read from the given excel file

Return:

1. dataFrame: A pandas[2] data frame object constructed from reading the excel file
2. ***allocateData(dataFrame,columnName)***

This function is used to convert a dataframe column to array for easier use. It discards any nan values.

Input parameters:

1. dataFrame: the dataframe
2. columnName: name of the column from where the data is to be retrieved into the array

Return:

1. dataArray: Array that holds the values of the specified column in the dataframe
2. ***calculate(dataFrame,columnNames)***

This function is used to calculate Mean, Variance and Standard Deviation of a given dataframe column. Numpy[1] has been used for calculation.

Input parameters:

1. dataFrame: the dataframe
2. columnName: name of the column from where the data is to be retrieved into the array

Return:

1. resultArray: Array that holds the mean, variance and standard deviation of the given dataframe column.
2. ***calculateCovariance(dataFrame, columnNames)***

This function is used to calculate the covariance matrix of given data using numpy’s[1] inbuilt function cov()

Input parameters:

1. dataFrame: the dataframe
2. columnNames: It’s a list of lists. It contains a list of the various columns of the dataframe and each column has a list of values. In our case, we have 4 different columns and each of it has 49 values.

Return:

1. covMat: The covariance matrix of the given dataframe columns.
2. ***calculateCorrelationCoeff(dataFrame, columnNames)***

This function is used to calculate the correlation coefficients matrix of given data using numpy’s[1] inbuilt function corrcoef()

Input parameters:

1. dataFrame: the dataframe
2. columnNames: It’s a list of lists. It contains a list of the various columns of the dataframe and each column has a list of values. In our case, we have 4 different columns and each of it has 49 values.

Return:

1. corrcoeffMat: The correlation coefficients matrix of the given dataframe columns.
2. ***calculateLikelihood(dataFrame, columnNames)***

This function is used to calculate the log likelihood of the given data assuming each of the variables/features are mutually independent.

The **stdist[]** is standard distribution array which stores mean, variance and standard deviation of each dataframe column.

Since, this calculation is considering the features to be mutually independent, we calculate the log likelihood of each sample ***i*** in the training set by simply adding the log likelihoods of each of the values ***j*** in the ***i*** th training sample.

The logpdf function from scipy.stats[3] has been used for calculation of loglikelihood.

Input parameters:

1. dataFrame: the dataframe
2. columnNames: It’s a list of lists. It contains a list of the various columns of the dataframe and each column has a list of values. In our case, we have 4 different columns and each of it has 49 values.

Return:

1. logLikelihood: The log likelihood of the given data assuming each of the variables/features are mutually independent.
2. ***calculateMultivariateLikelihood(dataFrame, columnNames)***

This function is used to calculate the log likelihood of the given data assuming there is dependency among the variables/features.

**dArray[]** stores the columns of the dataframes in the form of an array using the *allocateData()* function **stdist[]** is standard distribution array which stores mean, variance and standard deviation of each dataframe column.

mean extracts the means of all the columns/features from **stdist[]**

The covariance of the data is stored in **cov**

for every training sample ***i,*** we extract the desired column/feature values in variable ***data.***

The loglikelihood of data is calculated using logpdf function from scipy.stats.multivariate\_normal[3] and added to net likelihood which is stored in loglikelihood.

Input parameters:

1. dataFrame: the dataframe
2. columnNames: It’s a list of lists. It contains a list of the various columns of the dataframe and each column has a list of values. In our case, we have 4 different columns and each of it has 49 values.

Return:

1. logLikelihood: The log likelihood of the given data assuming there is dependency among the variables/features.

We have kept the code modular and created a function for every requirement. Nothing is hard coded and the number of features/columns in the data set used can be changed and yet the code works. So this can be extended in future for extension of this project. The function plotBarGraph, correlationMatrixPlot, and scatterPlotMatrix are commented within the code. We can uncomment it to visualize data.

**3. Results**

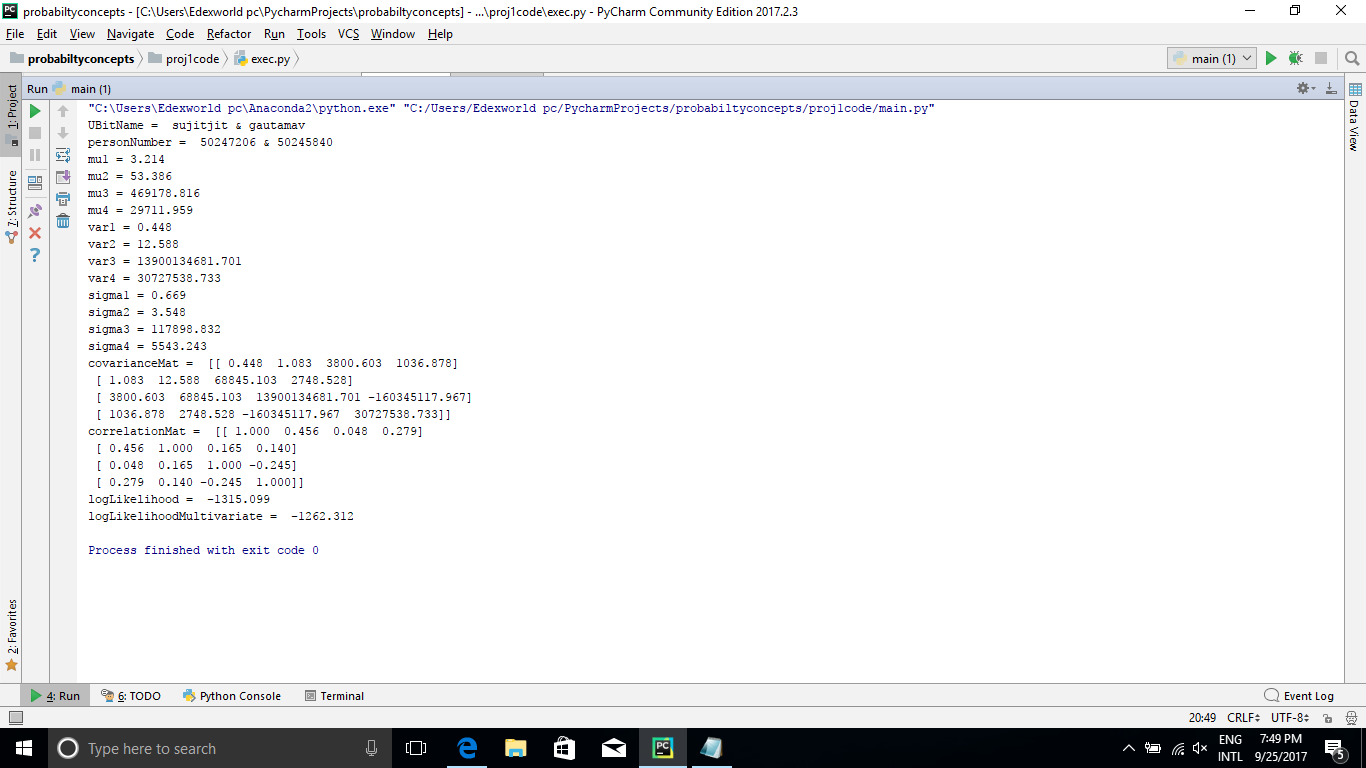


Fig 1.1 Implementation Snapshot

**4. Observation & Conclusion**

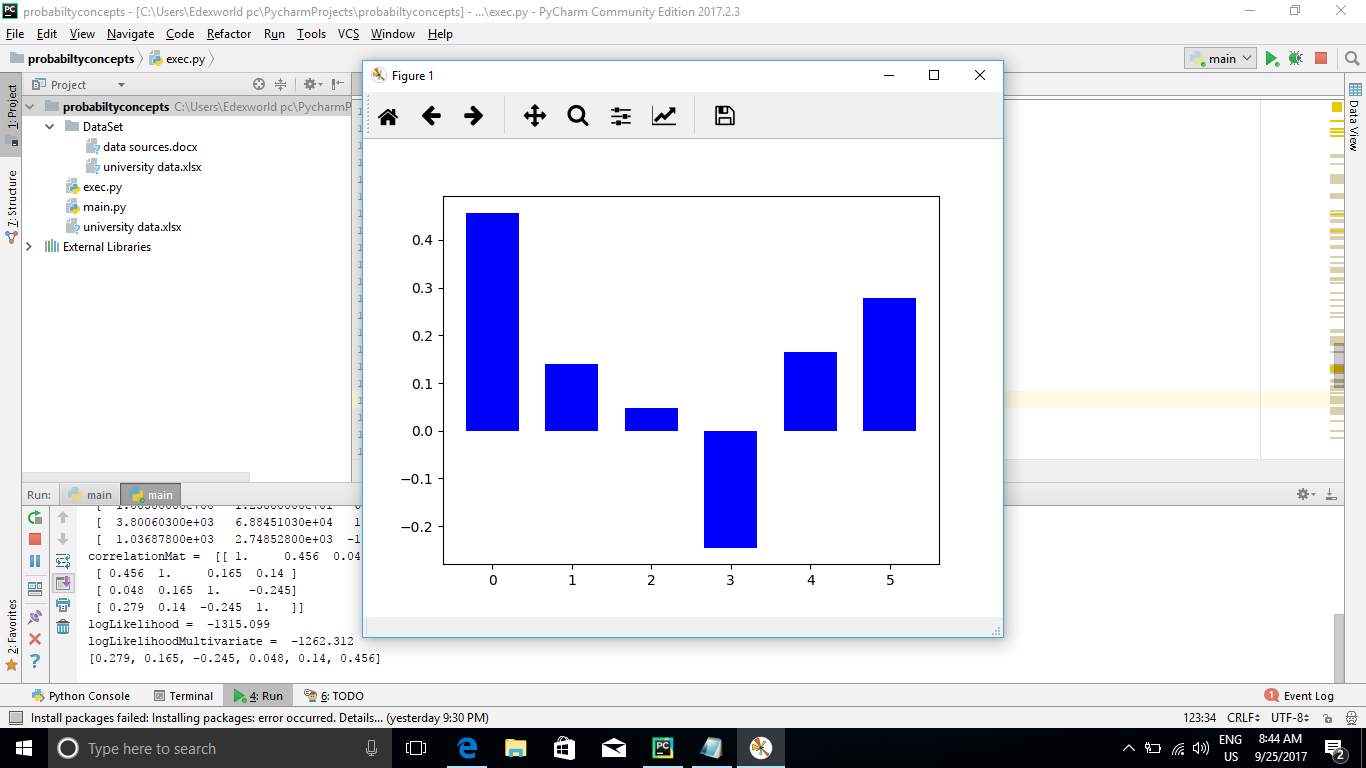


Fig 1.1 Bar Graph

The graph in the Fig 1.1 can be labelled as below:

["(CS Score, Tuition)","(Research Overhead, Admin Base Pay)","(Admin Base Pay - Tuition)",

"(CS Score - Admin Base Pay)","(Research Overhead - Tuition)","(CS Score – Research OverHead)"]

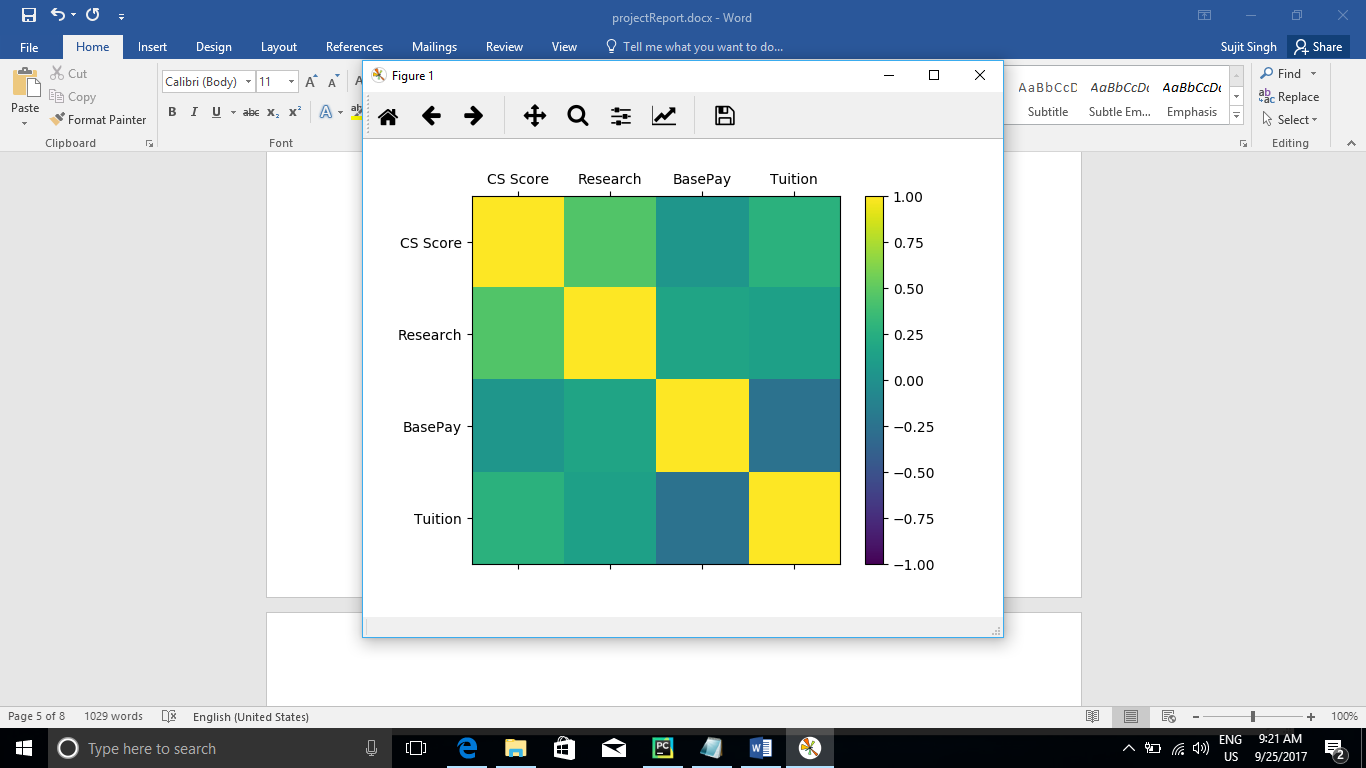


Fig 1.2 Correlation Matrix Plot

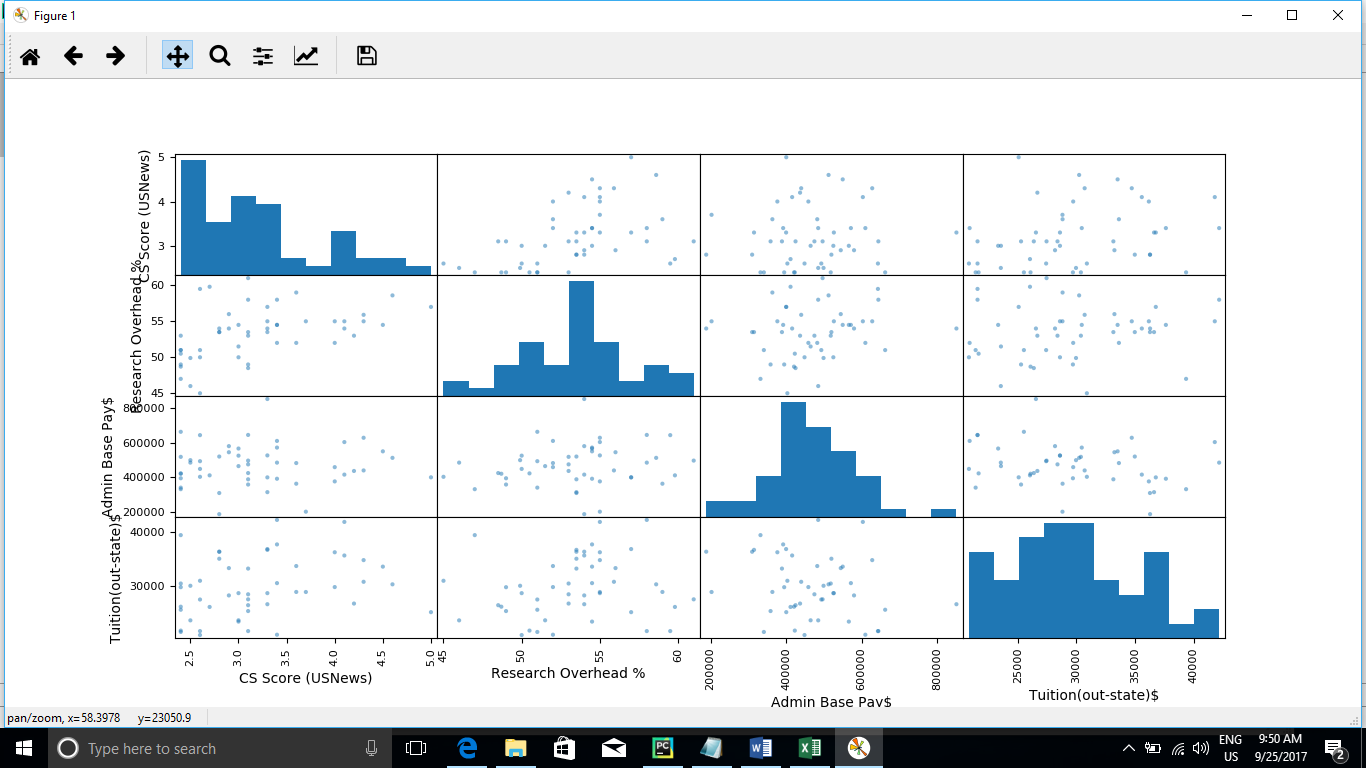


Fig 1.3 Pairwise Scatter Plot Matrix

From the above graph we can say that the pair (CS Score, Tuition) is highly correlated, whereas (Admin Base Pay - Tuition) is least correlated. On the other hand, (CS Score - Admin Base Pay) is negatively correlated. The correlation of the diagonal elements in the correlation matrix is highest i.e. 1. By comparing log-likelihood of independent variable and multivariate function, the value of multivariate function seems to be more. Hence, the multivariate implementation of log-likelihood improves the likelihood of occurrence.

**5. References:**

[1] Numpy tutorial - <https://docs.scipy.org/doc/numpy-dev/user/quickstart.html>

[2] Pandas tutorial - <https://pandas.pydata.org/pandas-docs/stable/tutorials.html>

[3] Scipy tutorial - <https://docs.scipy.org/doc/scipy/reference/tutorial/>

[4] MatPlotlib tutorial - https://matplotlib.org/users/pyplot\_tutorial.html