CS5590 APL - Python Programming

LAB2
Deadline: 3/13/2019

Hao Zhang
Peihao Fang
Ziqing Chen

I. Introduction

Our group ID is 3. The lab2 has 4 questions. Zhang Hao is mainly responsible for question 1 and 2, Fang Peihao is mainly responsible for question 3, and Chen Ziqing is mainly responsible for question 4.

II. Objectives

For the question 1:
We have two objectives:

First, we want to perform exploratory data analysis on the data set such as handling null values, removing the features not correlated to the target class and encoding the categorical features.

Then we want to apply the three classification algorithms Naïve Baye' s, SVM and KNN on the chosen data set and report which classifier gives better result.

For the question 2:
We have two objectives based on K-means method.
Firstly, we need to Apply K-means on the dataset, visualize the clusters using matplotlib or seaborn and find which K is the best using the elbow method.
After that, we Evaluate with silhouette score or other scores relevant for unsupervised approaches.
For the question 3:
We have 8 objectives
We write a program in which take an Input file and implement 8 functions.
a. Read the data from a file
b. Tokenize the text into words and apply lemmatization technique on each word.
c. Find all the trigrams for the words.
d. Extract the top 10 of the most repeated trigrams based on their count.
e. Go through the text in the file
f. Find all the sentences with the most repeated tri-grams
g. Extract those sentences and concatenate
h. Print the concatenated result
For the question 4:
Our objective is Creating Multiple Regression by choosing a dataset of your choice and Evaluating the model using RMSE and R2.

III. Approaches/Methods

Question 1:

We used pd.read_csv to read the dataset, then we used train.fillna(train.mean(), inplace=True) to handle the null values, after that we used train = train.drop(['MSZoning'], axis=1) to remove the features not correlated to the target class, then we used labelEncoder = preprocessing.LabelEncoder() to encode the categorical features, besides, we imported the SVC to fit the data(svm.fit(X_train, y_train)). We used print(metrics.confusion_matrix(expected, predicted_label)) to compare the predicted and expected values. We imported GaussianNB, and then Fit gaussian Naive Bayes model to the data. We imported import LogisticRegression to calculate the knn.

Question 2:

We used pd.read_csv to read the college dataset. Using apps, accept, enroll for clustering. ax.scatter(df.Apps, df["Accept"], df["Enroll"], c='blue', s=60)

Importing KMeans to apply K-Means, and using score = silhouette_score(df.iloc[:,[2,4]], kmeans.labels_, metric='euclidean') to calculate the score.

Using clusters = km.fit_predict(df.iloc[:,[2,4]]) to implement clustering.

Question 3:

1. Read content from a txt file
2. Use sentence tokenize and word tokenize to separate content into individual.
3. Apply lemmatization on each word and find trigrams
4. Extract top 10 the most repeated word base on counts by FreqDist
5. Use find method and concatenated string to localize word in sentence

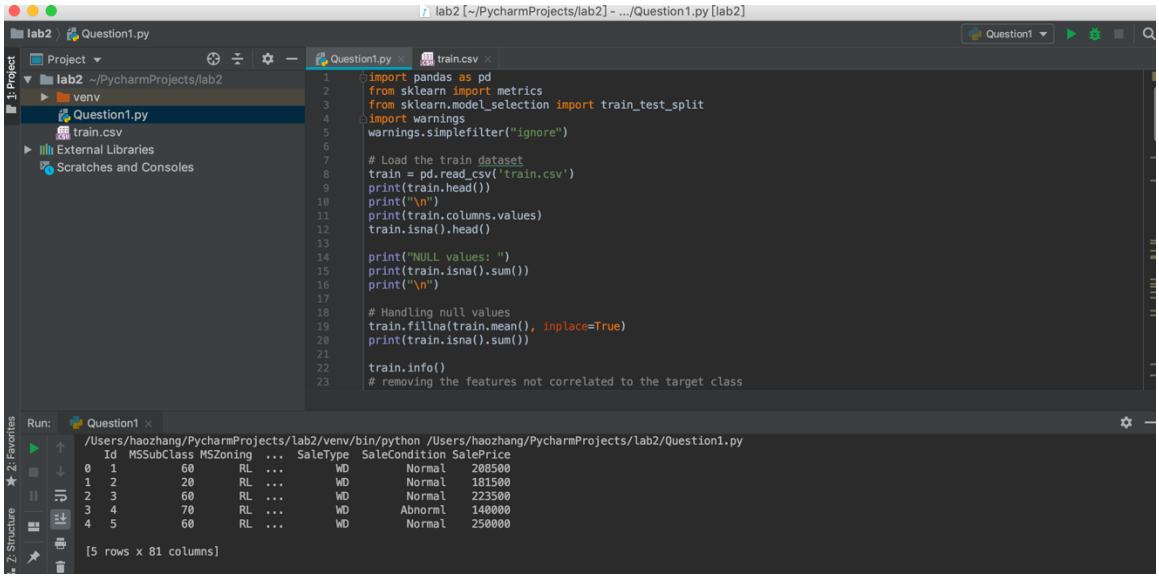
```
nltk.word_tokenize(i)
nltk.sent_tokenize(i)
le=WordNetLemmatizer()
ngrams(splitWord, 3):
wordFreq = FreqDist(trigramsOutput)
# Getting Most Common Words and Printing them - Will get the Counts from top
to least
top10 = wordFreq.most_common(10)
```

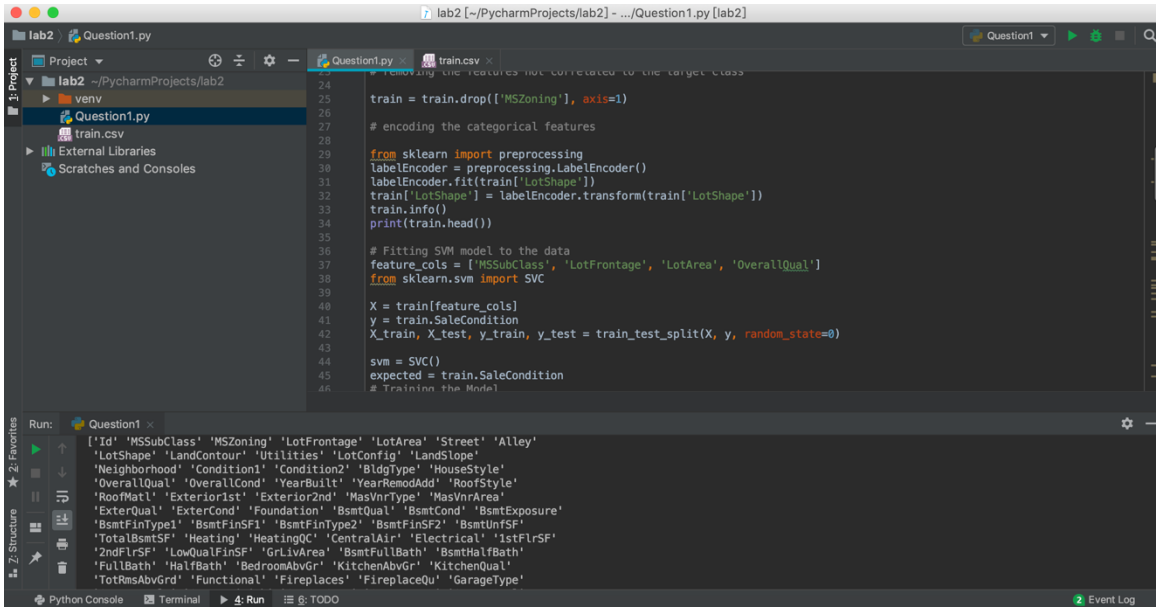
Question 4:

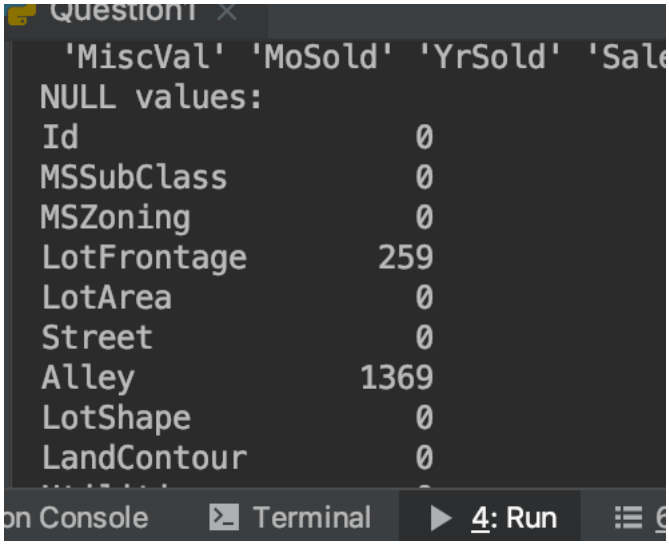
RMSE and R2

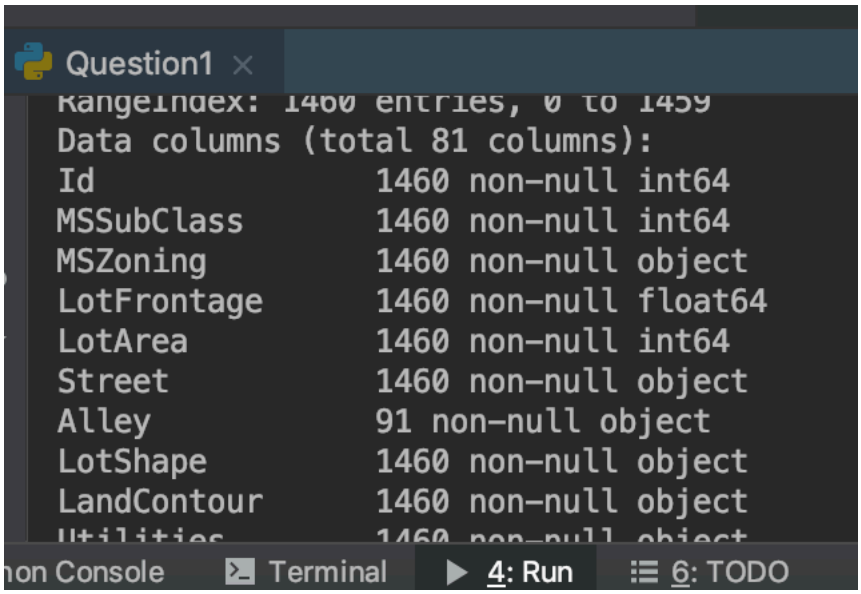
IV. Workflow

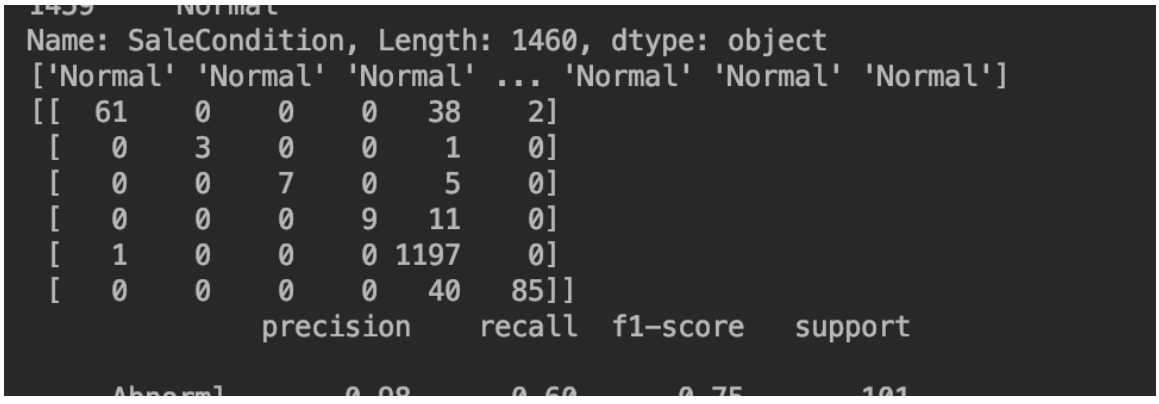
Question 1:


lab2 [~/PycharmProjects/lab2] - .../Question1.py [lab2]
lab2 Question1.py
Question1
Project
lab2 ~/PycharmProjects/lab2
venv
Question1.py
train.csv
External Libraries
Scratches and Consoles
Question1.py
train.csv
import pandas as pd
from sklearn import metrics
from sklearn.model_selection import train_test_split
import warnings
warnings.simplefilter("ignore")
# Load the train dataset
train = pd.read_csv('train.csv')
print(train.head())
print("\n")
print(train.columns.values)
train.isna().head()
print("NULL values: ")
print(train.isna().sum())
print("\n")
# Handling null values
train.fillna(train.mean(), inplace=True)
print(train.isna().sum())
train.info()
# removing the features not correlated to the target class
Run:
Question1
/Users/haozhang/PycharmProjects/lab2/venv/bin/python /Users/haozhang/PycharmProjects/lab2/Question1.py
Id MSSubClass MSZoning ... SaleType SaleCondition SalePrice
0 1 60 RL ... WD Normal 208500
1 2 20 RL ... WD Normal 181500
2 3 60 RL ... WD Normal 223500
3 4 70 RL ... WD Abnorml 140000
4 5 60 RL ... WD Normal 250000
[5 rows x 81 columns]


lab2 [~/PycharmProjects/lab2] - .../Question1.py [lab2]
lab2 Question1.py
Question1
Project
lab2 ~/PycharmProjects/lab2
venv
Question1.py
train.csv
External Libraries
Scratches and Consoles
Question1.py
train.csv
train = train.drop(['MSZoning'], axis=1)
# encoding the categorical features
from sklearn import preprocessing
labelEncoder = preprocessing.LabelEncoder()
labelEncoder.fit(train['LotShape'])
train['LotShape'] = labelEncoder.transform(train['LotShape'])
train.info()
print(train.head())
# Fitting SVM model to the data
feature_cols = ['MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual']
from sklearn.svm import SVC
X = train[feature_cols]
y = train.SaleCondition
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
svm = SVC()
expected = train.SaleCondition
Run:
Question1
['Id' 'MSSubClass' 'MSZoning' 'LotFrontage' 'LotArea' 'Street' 'Alley'
'LotShape' 'LandContour' 'Utilities' 'LotConfig' 'LandSlope'
'Neighborhood' 'Condition1' 'Condition2' 'BldgType' 'HouseStyle'
'OverallQual' 'OverallCond' 'YearBuilt' 'YearRemodAdd' 'RoofStyle'
'RoofMatl' 'Exterior1st' 'Exterior2nd' 'MasVnrType' 'MasVnrArea'
'ExterQual' 'ExterCond' 'Foundation' 'BsmtQual' 'BsmtCond' 'BsmtExposure'
'BsmtFinType1' 'BsmtFinSF1' 'BsmtFinType2' 'BsmtFinSF2' 'BsmtUnfSF'
'TotalBsmtSF' 'Heating' 'HeatingQC' 'CentralAir' 'Electrical' '1stFlrSF'
'2ndFlrSF' 'LowQualFinSF' 'GrLivArea' 'BsmtFullBath' 'BsmtHalfBath'
'FullBath' 'HalfBath' 'BedroomAbvGr' 'KitchenAbvGr' 'KitchenQual'
'TotRmsAbvGrd' 'Functional' 'Fireplaces' 'FireplaceQu' 'GarageType'
Python Console
Terminal
4: Run
6: TODO
Event Log


Question1
'MiscVal' 'MoSold' 'YrSold' 'Sale
NULL values:
Id 0
MSSubClass 0
MSZoning 0
LotFrontage 259
LotArea 0
Street 0
Alley 1369
LotShape 0
LandContour 0
on Console
Terminal
4: Run


Question1
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
Id 1460 non-null int64
MSSubClass 1460 non-null int64
MSZoning 1460 non-null object
LotFrontage 1460 non-null float64
LotArea 1460 non-null int64
Street 1460 non-null object
Alley 91 non-null object
LotShape 1460 non-null object
LandContour 1460 non-null object
non Console
Terminal
4: Run
6: TODO


Name: SaleCondition, Length: 1460, dtype: object
['Normal' 'Normal' 'Normal' ... 'Normal' 'Normal' 'Normal']
[[61 0 0 0 38 2]
[0 3 0 0 1 0]
[0 0 7 0 5 0]
[0 0 0 9 11 0]
[1 0 0 0 1197 0]
[0 0 0 0 40 85]]
precision recall f1-score support

```
Accuracy of SVM classifier : 0.79
              precision    recall  f1-score   support

     Abnorml       0.00      0.00      0.00       101
     AdjLand       0.00      0.00      0.00         4
      Alloca       0.00      0.00      0.00        12
      Family       0.00      0.00      0.00        20
      Normal       0.83      0.96      0.89      1198
```

Console | Terminal | 4: Run | 6: TODO

nstalled successfully: Installed packages: 'sklearn' (today 15:47)

Question1

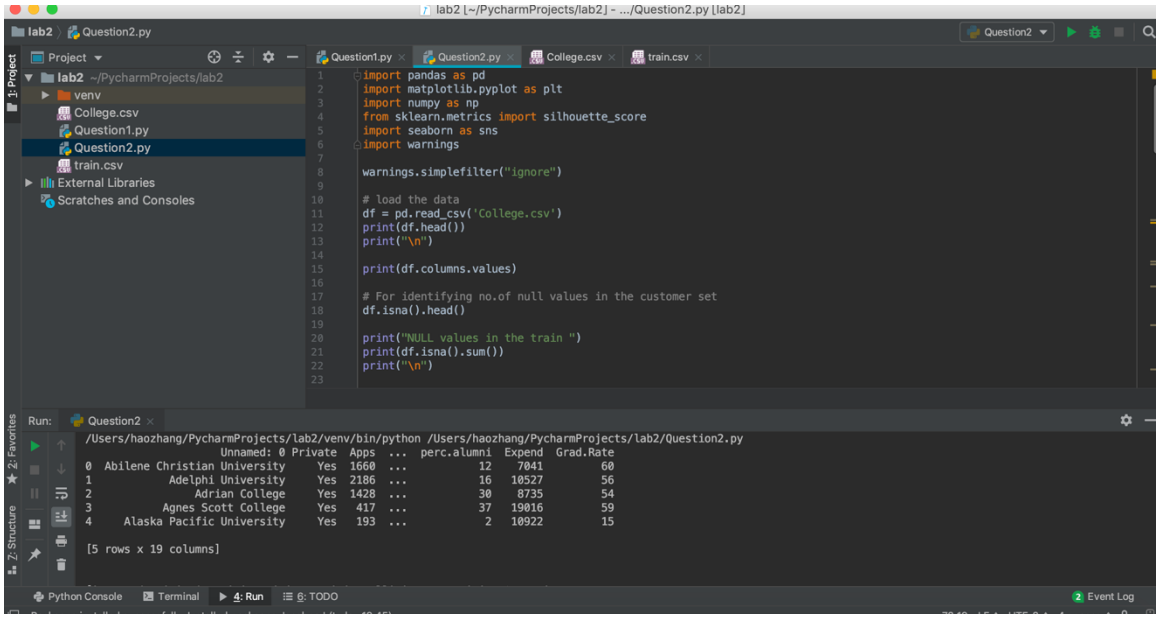
```
weighted avg       0.71      0.81      0.75      1460

[[   0    0    0    0   99    2]
 [   0    0    0    0    4    0]
 [   0    0    0    0   12    0]
 [   0    0    0    0   19    1]
 [   0    0    0    1 1156   41]
 [   0    0    0    0  104   21]]
accuracy using Gaussian navie bayes Model is  73.97260273972603
accuracy using knn Model is  0.8253424657534246

Process finished with exit code 0
```

```
Y_predicted = model.predict(X_test)

print("accuracy using Gaussian navie bayes Model is ", metrics.accuracy_score(Y_test, Y_predicted) * 100)

logreg = LogisticRegression()

# fit the model with data

logreg.fit(X, y)
logreg.predict(X)
y_pred = logreg.predict(X)
len(y_pred)

#knn
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X, y)
y_pred = knn.predict(X)
print("accuracy using knn Model is ",metrics.accuracy_score(y, y_pred))
```

```
model = GaussianNB()
model.fit(train[feature_cols], train.SaleCondition)
expected = train.SaleCondition            # Making Prediction based on X, Y
predicted = model.predict(train[feature_cols])


print(metrics.classification_report(expected, predicted))

print(metrics.confusion_matrix(expected, predicted))
X_train, X_test, Y_train, Y_test = train_test_split(train[feature_cols], train.SaleCondition, test_size=0.2, random_state=0

model.fit(X_train, Y_train)

Y_predicted = model.predict(X_test)

print("accuracy using Gaussian navie bayes Model is ", metrics.accuracy_score(Y_test, Y_predicted) * 100)
```

```
X = train[feature_cols]
y = train.SaleCondition
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

svm = SVC()
expected = train.SaleCondition
# Training the Model

svm.fit(X_train, y_train)
print(expected)

predicted_label = svm.predict(train[feature_cols])
print(predicted_label)
print(metrics.confusion_matrix(expected, predicted_label))
print(metrics.classification_report(expected, predicted_label))

# Evaluating the Model based
print('Accuracy of SVM classifier : {:.2f}'
     .format(svm.score(X_test, y_test)))

#calculating navie bayes model
```

Question 2:



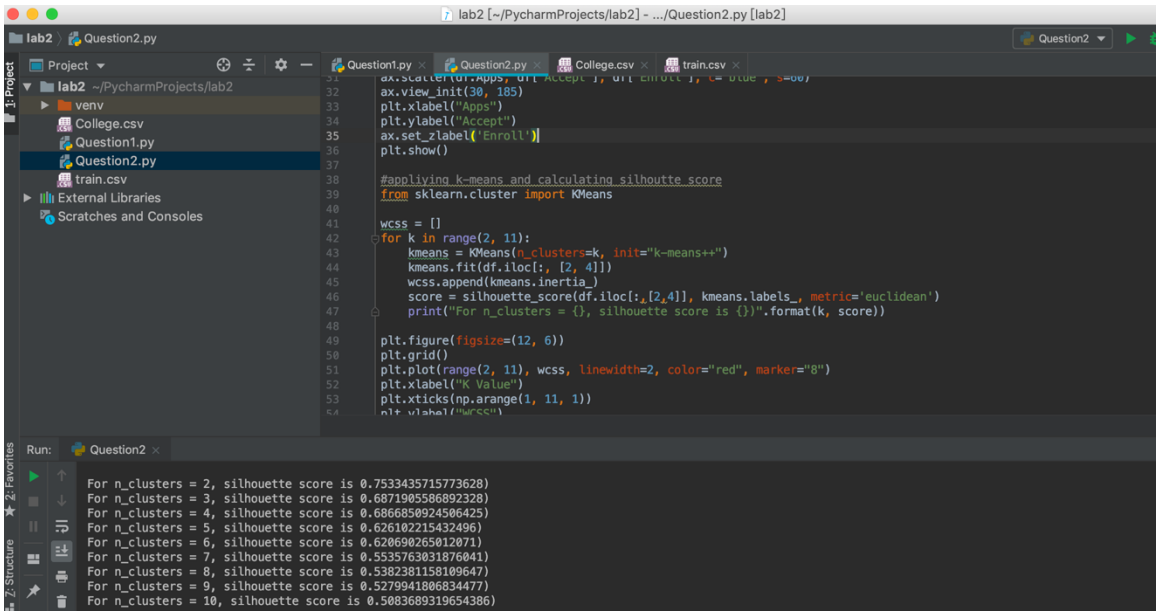
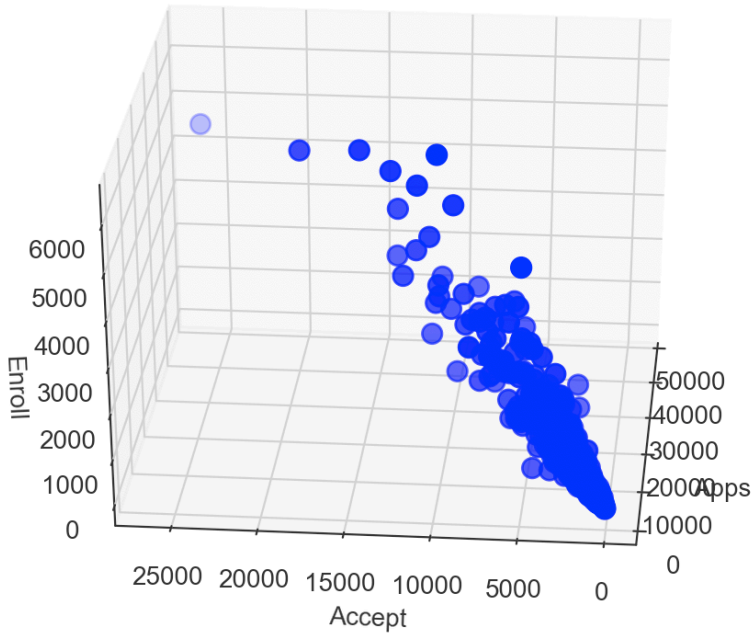

lab2 [~/PycharmProjects/lab2] - .../Question2.py [lab2]
Project
lab2 ~/PycharmProjects/lab2
venv
College.csv
Question1.py
Question2.py
train.csv
External Libraries
Scratches and Consoles
Question1.py
Question2.py
College.csv
train.csv
ax.view_init(30, 185)
plt.xlabel("Apps")
plt.ylabel("Accept")
ax.set_zlabel('Enroll')
plt.show()
#applying k-means and calculating silhoutte score
from sklearn.cluster import KMeans
wcss = []
for k in range(2, 11):
kmeans = KMeans(n_clusters=k, init="k-means++")
kmeans.fit(df.iloc[:, [2, 4]])
wcss.append(kmeans.inertia_)
score = silhouette_score(df.iloc[:,[2,4]], kmeans.labels_, metric='euclidean')
print("For n_clusters = {}, silhouette score is {})".format(k, score))
plt.figure(figsize=(12, 6))
plt.grid()
plt.plot(range(2, 11), wcss, linewidth=2, color="red", marker="8")
plt.xlabel("K Value")
plt.xticks(np.arange(1, 11, 1))
Run:
Question2
For n_clusters = 2, silhouette score is 0.7533435715773628)
For n_clusters = 3, silhouette score is 0.6871905586892328)
For n_clusters = 4, silhouette score is 0.6866850924506425)
For n_clusters = 5, silhouette score is 0.626102215432496)
For n_clusters = 6, silhouette score is 0.620690265012071)
For n_clusters = 7, silhouette score is 0.5535763031876041)
For n_clusters = 8, silhouette score is 0.5382381158109647)
For n_clusters = 9, silhouette score is 0.5279941806834477)
For n_clusters = 10, silhouette score is 0.5083689319654386)

Figure 1


6000
5000
4000
3000
2000
1000
0
Enroll
25000
20000
15000
10000
5000
0
Accept
50000
40000
30000
20000
10000
0
Apps

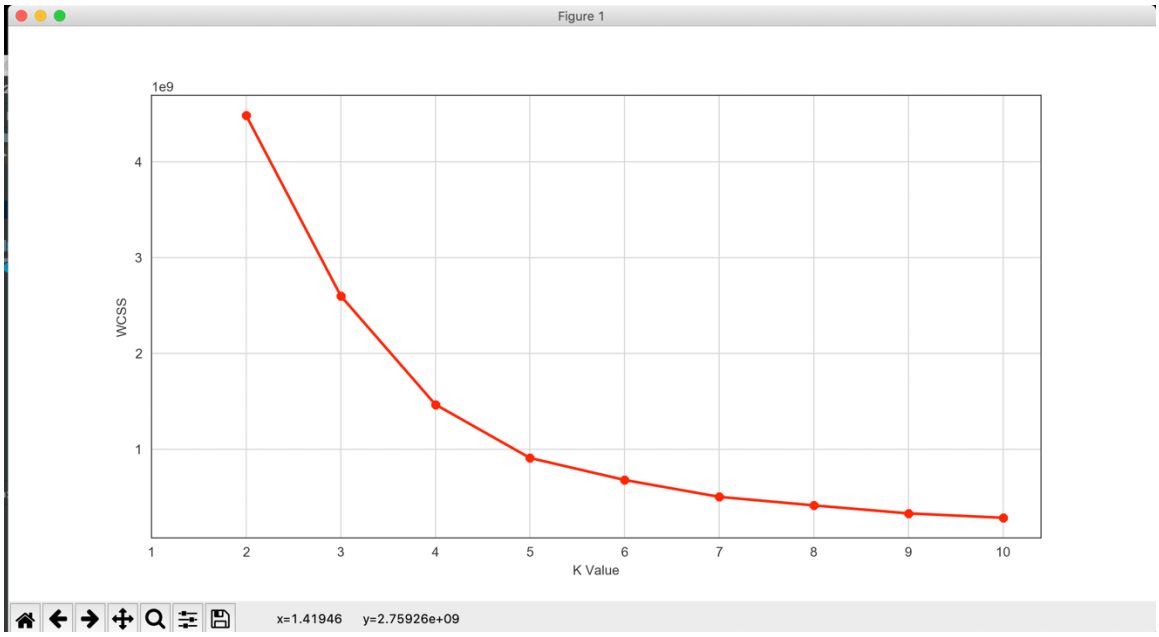
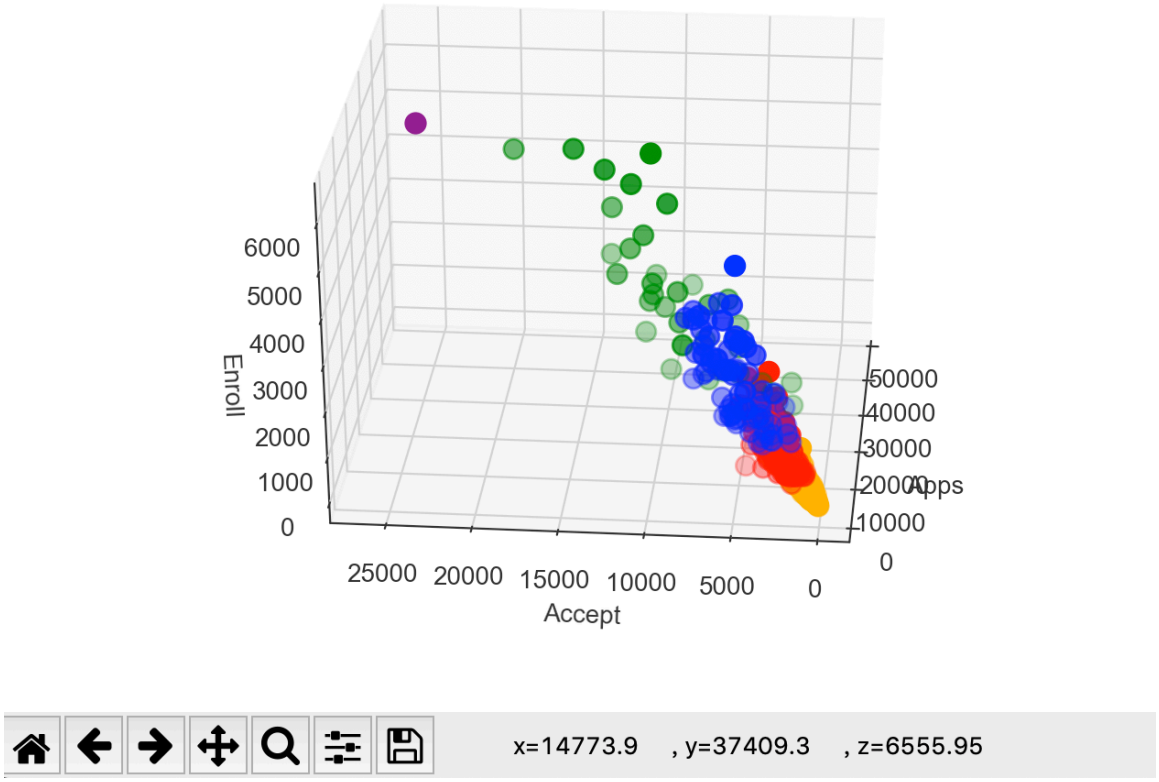


Figure 1



Question 3:

```
   1.        import requests
from bs4 import BeautifulSoup
import nltk
from nltk import PorterStemmer
from nltk import WordNetLemmatizer
from nltk import wordpunct_tokenize, pos_tag, ne_chunk
from nltk import ngrams, FreqDist
from collections import Counter
le=WordNetLemmatizer()
#a
file=open("nlp_input.txt",encoding="utf8", errors='ignore')
fileToken=open("Token.txt","w+")
filele=open("Lemmatizer.txt","w+")
fileTri=open("TRI.text","w+")
trigramsOutput = []
n=0

for i in file.readlines():
    n=n+1
#b Tokenize the text into words and apply lemmatization
    splitWord = nltk.word_tokenize(i)
    splitSen=nltk.sent_tokenize(i)

    for m in ngrams(splitWord, 3):
        trigramsOutput.append(m)

    filele.write('[')
    for j in splitWord:
        l = le.lemmatize(j)
        filele.write(l)
        filele.write(',')
    fileToken.write(str(splitWord))
    filele.write(']')

file.close()
fileToken.close()
filele.close()
fileTri.write(str(trigramsOutput))
fileTri.close()
wordFreq = FreqDist(trigramsOutput)
# Getting Most Common Words and Printing them - Will get the Counts from
```

```
top to least
top10 = wordFreq.most_common(10)
print("Top 10 triGrams : \n", top10)

file=open("nlp_input.txt",encoding="utf8", errors='ignore')
x=file.read()
splitSen=nltk.sent_tokenize(x)

concatenate=[]
for ((e,d,f),len) in top10:
    a=e+" "+d+" "+f
    for sentence in splitSen:
        if(sentence.find(a)>0):
            concatenate.append(sentence)
print(concatenate)
```

Question 4:

Select data set->Clean the data set with EDA->Evaluate the model using RMSE and R2 ->Watch the data change

```
import warnings
warnings.simplefilter("ignore")
import pandas as pd
import warnings
warnings.simplefilter("ignore")
# Importing the dataset
dataset = pd.read_csv('dataset4.csv')
dataset.describe()
dataset["Insulin"].value_counts()
dataset.groupby(['Insulin', 'BMI']).mean()

dataset = dataset.fillna(dataset.mean())



#X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 2].values
X = dataset.drop(['Pregnancies','Insulin','SkinThickness','BMI'],axis=1)
#dataset = dataset.fillna(dataset.mean())



#df = df_train.drop(['Summary','Daily Summary'],axis=1)

#X = pd.get_dummies(X, columns=["Precip Type"])
#before EDA
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
model = regressor.fit(X_train, y_train)
```

```
from sklearn.metrics import mean_squared_error, r2_score
print("Variance score: %.2f" % r2_score(y_test,y_pred))
print("Mean squared error: %.2f" % mean_squared_error(y_test,y_pred))

# Encoding categorical data
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder = LabelEncoder()
X.values[:, 1] = labelencoder.fit_transform(X.values[:, 1])
onehotencoder = OneHotEncoder(categorical_features = [3])
X = onehotencoder.fit_transform(X).toarray()
corr = dataset.corr()

print (corr['Insulin'].sort_values(ascending=False)[:3], '\n')
print (corr['Insulin'].sort_values(ascending=False)[-3:])

# Avoiding the Dummy Variable Trap
X = X[:, 1:]
```

```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)



# Fitting Multiple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
model = regressor.fit(X_train, y_train)

# Predicting the Test set results
y_pred = regressor.predict(X_test)


#Evaluating the model

dataset = dataset.fillna(dataset.mean())
#dataset_numeric = dataset.filter(like="_N", axis=1)
from sklearn.metrics import mean_squared_error, r2_score
print("Variance score: %.2f" % r2_score(y_test,y_pred))
print("Mean squared error: %.2f" % mean_squared_error(y_test,y_pred))
```

The result before EDA

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
0	1	0	3	1	22.0	1	0	7.2500
1	2	1	1	0	38.0	1	0	71.2833
2	3	1	3	0	26.0	0	0	7.9250
3	4	1	1	0	35.0	1	0	53.1000
4	5	0	3	1	35.0	0	0	8.0500

```
Variance score: 0.40
Mean squared error: 0.14
```

The result after EDA

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
0	1	0	3	1	22.0	1	0	7.2500
1	2	1	1	0	38.0	1	0	71.2833
2	3	1	3	0	26.0	0	0	7.9250
3	4	1	1	0	35.0	1	0	53.1000
4	5	0	3	1	35.0	0	0	8.0500

Variance score: 0.19

Mean squared error: 0.19

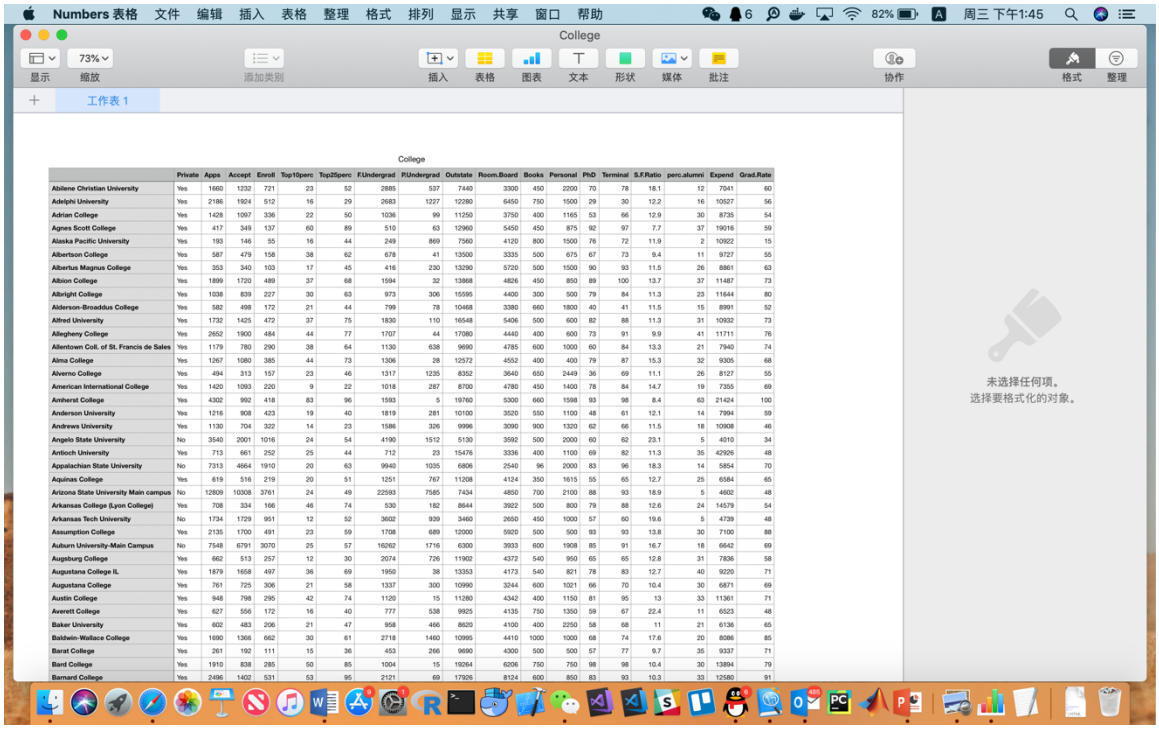
V. Datasets (if applicable)

Question 1:

The dataset is train.csv

Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	BldgType	HouseStyle	OverallQual	OverallCond	YearBuilt	YearRemodAdd	RoofStyle	R
1	60	RL	65	8450	Pave	NA	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	Norm	1Fam	2Story	7	5	2003	2003	Gable	C
2	20	RL	80	9600	Pave	NA	Reg	Lvl	AllPub	FR2	Gtl	Veenker	Feedr	Norm	1Fam	1Story	6	8	1976	1976	Gable	C
3	60	RL	68	11250	Pave	NA	IR1	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	Norm	1Fam	2Story	7	5	2001	2002	Gable	C
4	70	RL	60	9550	Pave	NA	IR1	Lvl	AllPub	Corner	Gtl	Crawfor	Norm	Norm	1Fam	2Story	7	5	1915	1970	Gable	C
5	60	RL	84	14260	Pave	NA	IR1	Lvl	AllPub	FR2	Gtl	NoRidge	Norm	Norm	1Fam	2Story	8	5	2000	2000	Gable	C
6	50	RL	85	14115	Pave	NA	IR1	Lvl	AllPub	Inside	Gtl	Mitchel	Norm	Norm	1Fam	1.5Fin	5	5	1993	1995	Gable	C
7	20	RL	75	10084	Pave	NA	Reg	Lvl	AllPub	Inside	Gtl	Somerst	Norm	Norm	1Fam	1Story	8	5	2004	2005	Gable	C
8	60	RL	NA	10382	Pave	NA	IR1	Lvl	AllPub	Corner	Gtl	NWAmes	PosN	Norm	1Fam	2Story	7	6	1973	1973	Gable	C
9	50	RM	51	6120	Pave	NA	Reg	Lvl	AllPub	Inside	Gtl	OldTown	Artery	Norm	1Fam	1.5Fin	7	5	1931	1950	Gable	C
10	190	RL	50	7420	Pave	NA	Reg	Lvl	AllPub	Corner	Gtl	BrkSide	Artery	Artery	2fmCon	1.5Unf	5	6	1939	1950	Gable	C
11	20	RL	70	11200	Pave	NA	Reg	Lvl	AllPub	Inside	Gtl	Sawyer	Norm	Norm	1Fam	1Story	5	5	1965	1965	Hip	C
12	60	RL	85	11924	Pave	NA	IR1	Lvl	AllPub	Inside	Gtl	NridgHt	Norm	Norm	1Fam	2Story	9	5	2005	2006	Hip	C
13	20	RL	NA	12968	Pave	NA	IR2	Lvl	AllPub	Inside	Gtl	Sawyer	Norm	Norm	1Fam	1Story	5	6	1962	1962	Hip	C
14	20	RL	91	10652	Pave	NA	IR1	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	Norm	1Fam	1Story	7	5	2006	2007	Gable	C
15	20	RL	NA	10920	Pave	NA	IR1	Lvl	AllPub	Corner	Gtl	NAmes	Norm	Norm	1Fam	1Story	6	5	1960	1960	Hip	C
16	45	RM	51	6120	Pave	NA	Reg	Lvl	AllPub	Corner	Gtl	BrkSide	Norm	Norm	1Fam	1.5Unf	7	8	1929	2001	Gable	C
17	20	RL	NA	11241	Pave	NA	IR1	Lvl	AllPub	CulDSac	Gtl	NAmes	Norm	Norm	1Fam	1Story	6	7	1970	1970	Gable	C
18	90	RL	72	10791	Pave	NA	Reg	Lvl	AllPub	Inside	Gtl	Sawyer	Norm	Norm	Duplex	1Story	4	5	1967	1967	Gable	C
19	20	RL	66	13695	Pave	NA	Reg	Lvl	AllPub	Inside	Gtl	SawyerW	RRAe	Norm	1Fam	1Story	5	5	2004	2004	Gable	C
20	20	RL	70	7560	Pave	NA	Reg	Lvl	AllPub	Inside	Gtl	NAmes	Norm	Norm	1Fam	1Story	5	6	1958	1965	Hip	C
21	60	RL	101	14215	Pave	NA	IR1	Lvl	AllPub	Corner	Gtl	NridgHt	Norm	Norm	1Fam	2Story	8	5	2005	2006	Gable	C
22	45	RM	57	7449	Pave	Grvl	Reg	Bnk	AllPub	Inside	Gtl	IDOTRR	Norm	Norm	1Fam	1.5Unf	7	7	1930	1950	Gable	C
23	20	RL	75	9742	Pave	NA	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	Norm	1Fam	1Story	8	5	2002	2002	Hip	C
24	120	RM	44	4224	Pave	NA	Reg	Lvl	AllPub	Inside	Gtl	MeadowV	Norm	Norm	TwnhsE	1Story	5	7	1976	1976	Gable	C
25	20	RL	NA	8246	Pave	NA	IR1	Lvl	AllPub	Inside	Gtl	Sawyer	Norm	Norm	1Fam	1Story	5	8	1968	2001	Gable	C
26	20	RL	110	14230	Pave	NA	Reg	Lvl	AllPub	Corner	Gtl	NridgHt	Norm	Norm	1Fam	1Story	8	5	2007	2007	Gable	C
27	20	RL	60	7200	Pave	NA	Reg	Lvl	AllPub	Corner	Gtl	NAmes	Norm	Norm	1Fam	1Story	5	7	1951	2000	Gable	C
28	20	RL	98	11478	Pave	NA	Reg	Lvl	AllPub	Inside	Gtl	NridgHt	Norm	Norm	1Fam	1Story	8	5	2007	2008	Gable	C
29	20	RL	47	16321	Pave	NA	IR1	Lvl	AllPub	CulDSac	Gtl	NAmes	Norm	Norm	1Fam	1Story	5	6	1957	1997	Gable	C
30	30	RM	60	6324	Pave	NA	IR1	Lvl	AllPub	Inside	Gtl	BrkSide	Feedr	RRNn	1Fam	1Story	4	6	1927	1950	Gable	C
31	70	C (all)	50	8500	Pave	Pave	Reg	Lvl	AllPub	Inside	Gtl	IDOTRR	Feedr	Norm	1Fam	2Story	4	4	1920	1950	Gambrel	C
32	20	RL	NA	8544	Pave	NA	IR1	Lvl	AllPub	CulDSac	Gtl	Sawyer	Norm	Norm	1Fam	1Story	5	6	1966	2006	Gable	C
33	20	RL	85	11049	Pave	NA	Reg	Lvl	AllPub	Corner	Gtl	CollgCr	Norm	Norm	1Fam	1Story	8	5	2007	2007	Gable	C
34	20	RL	70	10552	Pave	NA	IR1	Lvl	AllPub	Inside	Gtl	NAmes	Norm	Norm	1Fam	1Story	5	5	1959	1959	Hip	C
35	120	RL	60	7313	Pave	NA	Reg	Lvl	AllPub	Inside	Gtl	NridgHt	Norm	Norm	TwnhsE	1Story	9	5	2005	2005	Hip	C
36	60	RL	108	13418	Pave	NA	Reg	Lvl	AllPub	Inside	Gtl	NridgHt	Norm	Norm	1Fam	2Story	8	5	2004	2005	Gable	C

Question 2 dataset:

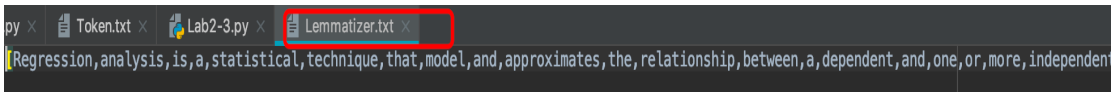


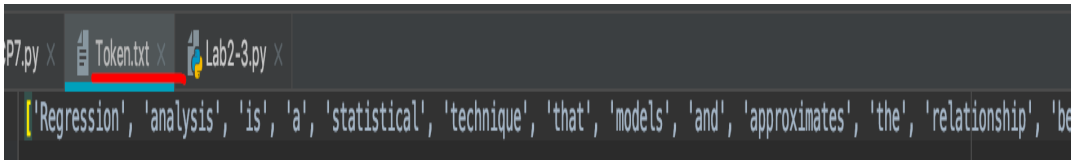
Question 3:
https://umkc.app.box.com/s/7by0f4540cdbdp3pm60h5fxxffefsvrw

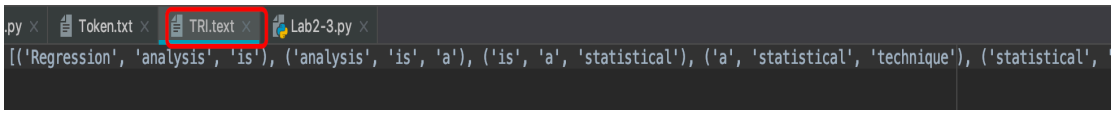
Question 4:
dataset4.

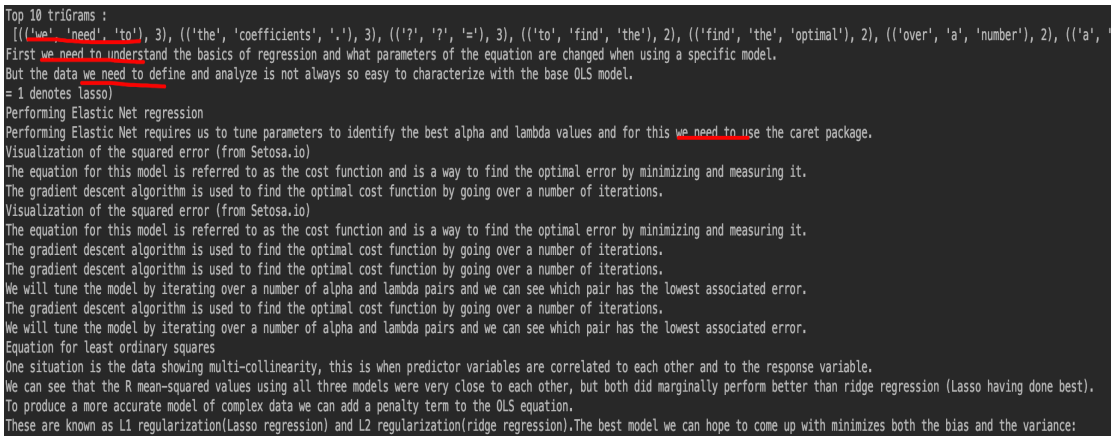
VII. Evaluation & Discussion

We don’ t find any problems about question 1. For question 2, we had an error about df.iloc(). Now, we know that df.iloc[ ] is primarily integer position based (from 0 to length-1 of the axis), but may also be used with a boolean array.
Question 3:









The weakness of my output is that I don’ t format the output to look more clearly and pretty.

Question4:

We used the dataset which is for diabetes. Then we clean the data set with EDA. Splitting the dataset into the Training set and Test set. Fitting Multiple Linear Regression to the Training set. Predicting the Test set results. Last evaluating the model.