

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 Bayes, 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 `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]

Project: lab2 [~/PycharmProjects/lab2]

- venv
- Question1.py
- train.csv
- External Libraries
- Scratches and Consoles

```
1 import pandas as pd
2 from sklearn import metrics
3 from sklearn.model_selection import train_test_split
4 import warnings
5 warnings.simplefilter("ignore")
6
7 # Load the train dataset
8 train = pd.read_csv('train.csv')
9 print(train.head())
10 print("\n")
11 print(train.columns.values)
12 print(train.isna().head())
13
14 print("NULL values: ")
15 print(train.isna().sum())
16 print("\n")
17
18 # Handling null values
19 train.fillna(train.mean(), inplace=True)
20 print(train.isna().sum())
21
22 train.info()
23 # 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	Abnormal	148000
4	5	60	RL	...	WD	Normal	250000

[5 rows x 81 columns]

lab2 [~/PycharmProjects/lab2] - .../Question1.py [lab2]

Project: lab2 [~/PycharmProjects/lab2]

- venv
- Question1.py
- train.csv
- External Libraries
- Scratches and Consoles

```
23 # removing the features not correlated to the target class
24
25 train = train.drop(['MSZoning'], axis=1)
26
27 # encoding the categorical features
28
29 from sklearn import preprocessing
30 labelEncoder = preprocessing.LabelEncoder()
31 labelEncoder.fit(train['LotShape'])
32 train['LotShape'] = labelEncoder.transform(train['LotShape'])
33 train.info()
34 print(train.head())
35
36 # Fitting SVM model to the data
37 feature_cols = ['MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual']
38 from sklearn.svm import SVC
39
40 X = train[feature_cols]
41 y = train.SaleCondition
42 X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
43
44 svm = SVC()
45 expected = train.SaleCondition
46 # Training the Model
```

Run: Question1

['Id' 'MSSubClass' 'MSZoning' 'LotFrontage' 'LotArea' 'Street' 'Alley'
'LotShape' 'LandContour' 'Utilities' 'LotConfig' 'LandSlope'
'Neighborhood' 'Condition' '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 | Run | TODO | Event Log

```

Question1 x
'MiscVal' 'MoSold' 'YrSold' 'Sale
NULL values:
Id          0
MSSubClass  0
MSZoning    0
LotFrontage 259
LotArea     0
Street      0
Alley       1369
LotShape    0
LandContour 0
Utilities   0

```

on Console Terminal ▶ 4: Run ≡ 6: TODO

```

Question1 x
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
Utilities   1460 non-null object

```

on Console Terminal ▶ 4: Run ≡ 6: TODO

```

1459 Normal
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

Abnormal    0.98    0.60    0.75      101

```

Accuracy of SVM classifier : 0.79

	precision	recall	f1-score	support
Abnormal	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
Partial	0.33	0.17	0.22	125

Console Terminal ▶ 4: Run ≡ 6: TODO

Installed 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  1156 41]
 [ 0  0  0  0 104 21]]
```

accuracy using Gaussian naive 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 naive 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 naive 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 naive bayes model

```

Question 2:

The screenshot shows the PyCharm IDE interface. The main editor displays a Python script named `Question2.py` with the following code:

```

1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import numpy as np
4 from sklearn.metrics import silhouette_score
5 import seaborn as sns
6 import warnings
7
8 warnings.simplefilter("ignore")
9
10 # load the data
11 df = pd.read_csv('College.csv')
12 print(df.head())
13 print("\n")
14
15 print(df.columns.values)
16
17 # For identifying no. of null values in the customer set
18 df.isna().head()
19
20 print("NULL values in the train ")
21 print(df.isna().sum())
22 print("\n")
23

```

The left sidebar shows the project structure for `lab2`, including files like `College.csv`, `Question1.py`, `Question2.py`, and `train.csv`.

The bottom panel shows the terminal output of the script:

```

Run: Question2 x
/Users/haozhang/PycharmProjects/lab2/venv/bin/python /Users/haozhang/PycharmProjects/lab2/Question2.py
  Unnamed: 0 Private Apps ... perc.alumni Expend Grad.Rate
0 Abilene Christian University Yes 1660 ... 12 7041 60
1 Adelphi University Yes 2186 ... 16 10527 56
2 Adrian College Yes 1428 ... 30 8735 54
3 Agnes Scott College Yes 417 ... 37 19016 59
4 Alaska Pacific University Yes 193 ... 2 10922 15
[5 rows x 19 columns]

```

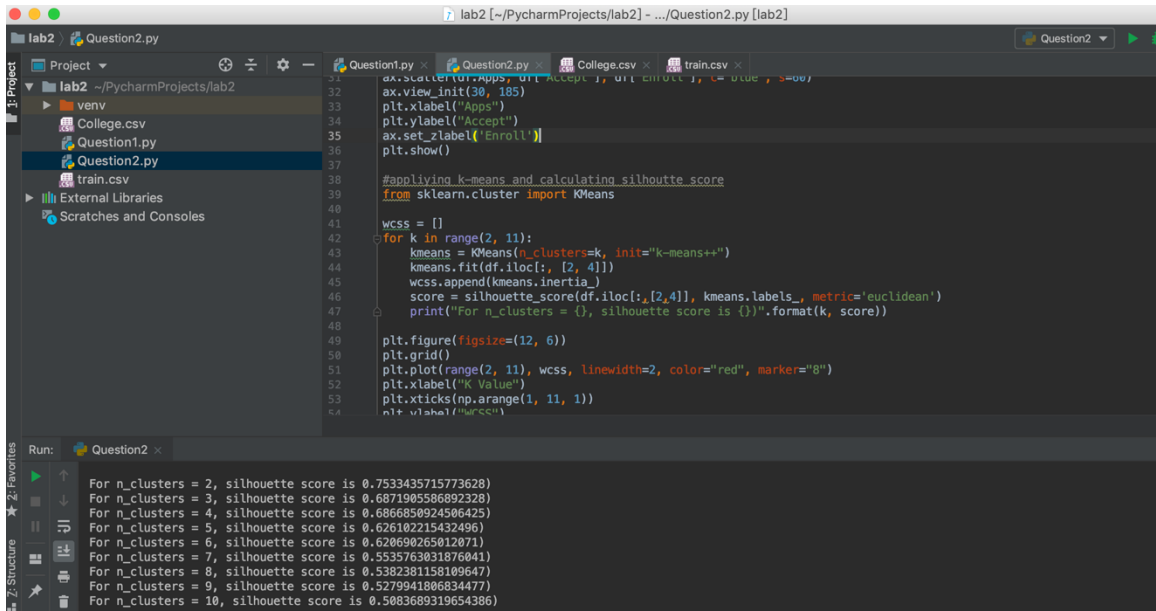
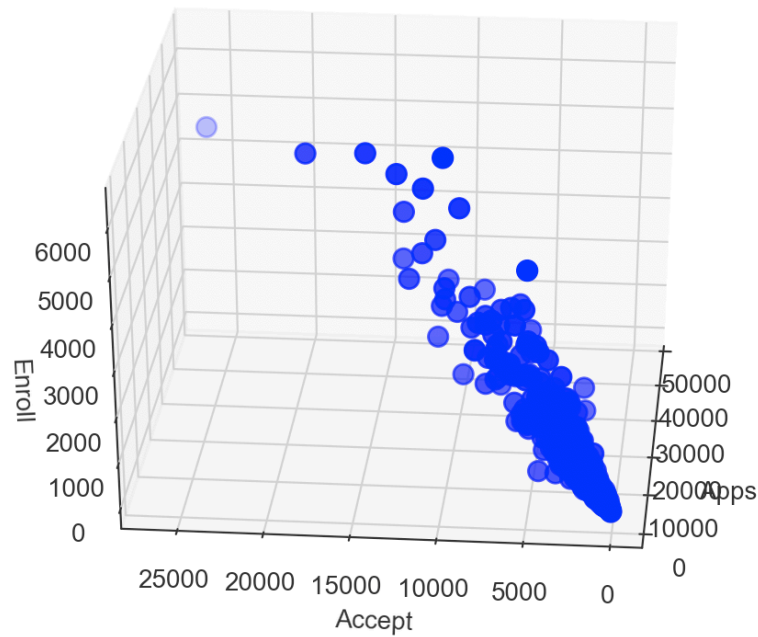
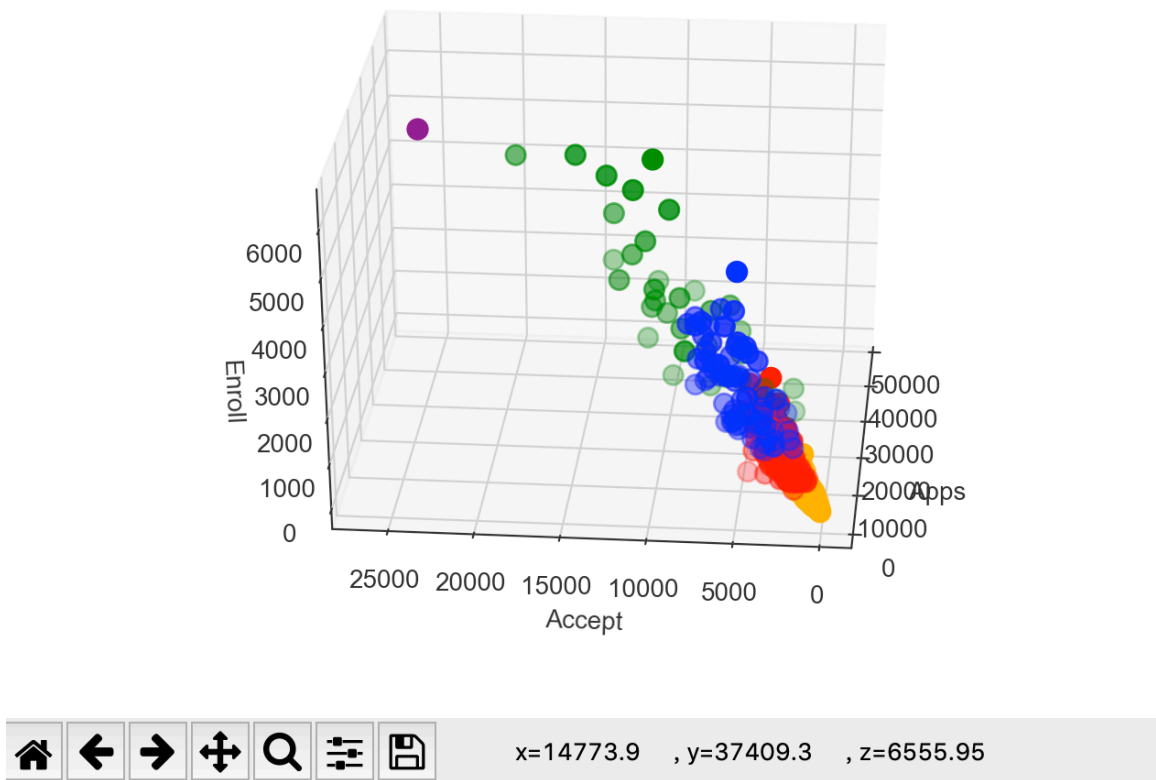
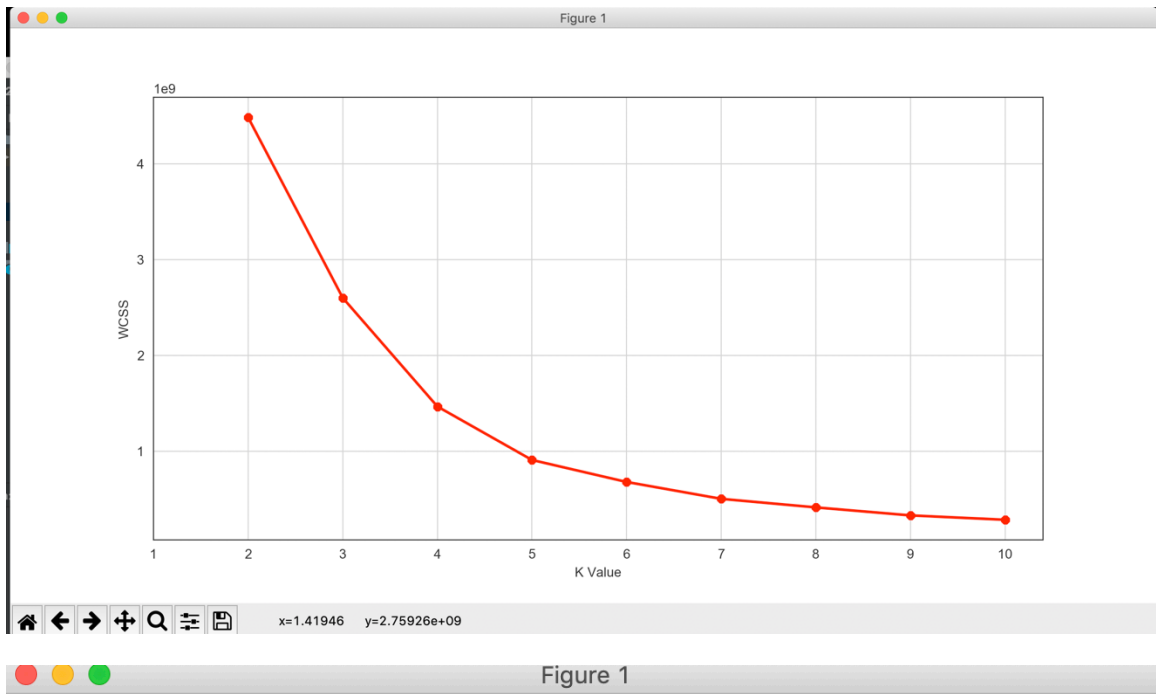


Figure 1





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

Question 2 dataset:

Numbers

Numbers 表格 文件 编辑 插入 表格 整理 格式 排列 显示 共享 窗口 帮助

6

82%

周三 下午1:45

显示 缩放

添加系列

插入 表格 图表 文本 形状 媒体 批注

协作

格式 整理

工作表 1

College																		
	Private	Apps	Accept	Enroll	Top25perc	Top25perc	FUndergrad	FUndergrad	Outstate	Room.Board	Books	Personal	PHD	Terminal	S.F.Ratio	per capita	Expend	Grad.Rate
Abilene Christian University	Yes	1660	1232	721	23	52	2885	537	7440	3300	450	2200	70	78	18.1	12	7041	60
Adelphi University	Yes	2186	1824	512	16	29	2683	1227	12280	6450	750	1500	29	30	12.2	16	10027	56
Adrian College	Yes	1428	1087	336	22	50	1038	99	11280	3750	450	1165	53	66	12.9	30	8735	54
Agnes Scott College	Yes	417	349	137	60	89	510	63	12960	5450	450	875	92	97	7.7	37	19016	59
Alaska Pacific University	Yes	193	146	55	16	44	249	869	7560	4120	800	1500	76	72	11.9	2	10822	15
Albertson College	Yes	587	479	158	38	62	678	41	13000	3335	500	675	67	73	9.4	11	9727	55
Albertus Magnus College	Yes	353	340	103	17	45	416	230	13280	5720	500	1500	90	93	11.5	26	8861	63
Albion College	Yes	1899	1720	489	37	68	1594	32	13888	4826	450	850	89	100	13.7	37	11487	73
Albright College	Yes	1038	829	227	30	63	973	306	10585	4400	300	500	79	84	11.3	23	11844	80
Alderson-Brooks College	Yes	582	498	172	21	44	799	78	10468	3380	600	1800	40	41	11.5	15	8991	52
Alfred University	Yes	1732	1425	472	37	75	1830	110	16548	5406	500	600	82	88	11.3	31	10832	73
Allegheny College	Yes	2652	1900	484	44	77	1707	44	17080	4440	400	600	73	91	9.9	41	11711	76
Allenstown Coll. of St. Francis de Sales	Yes	1179	780	290	38	64	1130	638	9980	4785	800	1000	60	84	13.3	21	7940	74
Alma College	Yes	1267	1080	385	44	73	1306	28	12572	4552	400	400	79	87	16.3	32	9305	68
Alverno College	Yes	494	313	157	23	40	1217	1235	8332	3640	600	2449	36	69	11.1	26	8137	55
American International College	Yes	1420	1093	220	9	22	1018	287	8700	4780	450	1400	78	84	14.7	19	7355	69
Anshatt College	Yes	4302	992	418	83	96	1093	5	18780	5300	660	1998	93	98	8.4	63	21424	100
Anderson University	Yes	1216	908	423	19	40	1819	281	10180	3520	550	1100	48	61	12.1	14	7994	59
Andrews University	Yes	1130	704	322	14	23	1588	326	9996	3090	800	1320	62	66	11.5	18	10908	46
Angelo State University	No	3540	2081	1016	24	54	4180	1612	9130	3962	500	2000	60	62	23.1	5	4910	34
Antioch University	Yes	713	661	252	25	44	712	23	15478	3336	400	1100	69	82	11.3	35	42926	48
Appalachian State University	No	7313	4664	1910	20	63	9940	1035	6806	2540	96	2000	83	96	18.3	14	5854	70
Aquinas College	Yes	819	516	219	20	51	1251	767	11208	4124	300	1615	55	65	12.7	25	6084	65
Arizona State University Main campus	No	12809	10308	3761	24	49	22593	7585	7434	4850	700	2100	88	93	18.9	5	4602	48
Arkansas College (Lyon College)	Yes	708	334	166	46	74	530	182	8544	2822	500	800	79	88	12.6	24	14379	54
Arkansas Tech University	No	1734	1129	501	12	52	3682	939	3480	2050	450	1000	57	60	19.6	5	4739	48
Assumption College	Yes	2135	1700	491	23	59	1708	689	13500	5820	500	500	93	93	13.8	30	7100	88
Auburn University Main Campus	No	7548	6791	3070	25	57	16262	1716	6300	3933	600	1908	85	91	16.7	18	6642	69
Augsburg College	Yes	662	513	257	12	30	2074	726	11902	4372	540	950	65	65	12.8	31	7836	58
Augustana College IL	Yes	1879	1658	487	36	69	1950	38	13353	4173	540	821	78	83	12.7	40	9220	71
Augustana College	Yes	761	725	306	21	58	1337	300	10990	3244	600	1021	66	70	16.4	30	6871	69
Austin College	Yes	848	788	296	42	74	1120	15	11380	4342	400	1160	81	95	13	32	11381	71
Averett College	Yes	627	556	172	16	40	777	538	9825	4135	750	1350	59	67	22.4	11	6523	48
Baker University	Yes	602	483	206	21	47	958	466	8820	4100	400	2250	58	68	11	21	6136	65
Baldwin-Wallace College	Yes	1680	1366	662	30	61	2718	1460	10995	4410	1000	1000	68	74	17.6	20	8086	85
Barat College	Yes	261	192	111	15	36	453	266	9890	4300	500	500	57	77	9.7	35	9337	71
Barst College	Yes	1910	828	285	50	85	1004	15	16204	6206	750	750	98	98	10.4	30	13994	79
Bemard College	Yes	2452	1402	521	63	95	2121	60	17925	8714	600	850	83	93	10.3	23	12580	91

未选择任何项。
选择要格式化的对象。

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:

py x Token.txt x Lab2-3.py x Lemmatizer.txt x
 Regression, analysis, is, a, statistical, technique, that, model, and, approximates, the, relationship, between, a, dependent, and, one, or, more, independent

py x Token.txt x Lab2-3.py x
 ['Regression', 'analysis', 'is', 'a', 'statistical', 'technique', 'that', 'models', 'and', 'approximates', 'the', 'relationship', 'be

```
py x Token.txt x TRI.text x Lab2-3.py x
[['Regression', 'analysis', 'is'], ('analysis', 'is', 'a'), ('is', 'a', 'statistical'), ('a', 'statistical', 'technique'), ('statistical',
```

Top 10 triGrams :
[[('we', 'need', 'to'), 3], (('the', 'coefficients', '.'), 3), (('?', '?', '='), 3), (('to', 'find', 'the'), 2), (('find', 'the', 'optimal'), 2), (('over', 'a', 'number'), 2), (('a', 'number', 'of'), 2), (('the', 'data', 'we'), 2), (('we', 'need', 'to'), 2), (('the', 'data', 'we'), 2)]
First we need to understand the basics of regression and what parameters of the equation are changed when using a specific model.
But the data we need to define and analyze is not always

Top 10 triGrams :
[[('we', 'need', 'to'), 3], (('the', 'coefficients', '.'), 3), (('?', '?', '='), 3), (('to', 'find', 'the'), 2), (('find', 'the', 'optimal'), 2), (('over', 'a', 'number'), 2), (('a', 'number', 'of'), 2), (('the', 'data', 'we'), 2), (('we', 'need', 'to'), 2), (('the', 'data', 'we'), 2)]
First we need to understand the basics of regression and what parameters of the equation are changed when using a specific model.
But the data we need to define and analyze is not always so easy to characterize with the base OLS model.
= 1 denotes (lasso)
Performing Elastic Net regression
Performing Elastic Net requires us to tune parameters to identify the best alpha and lambda values and for this we need to use the caret package.
Visualization of the squared error (from Setosa.io)
The equation for this model is referred to as the cost function and is a way to find the optimal error by minimizing and measuring it.
The gradient descent algorithm is used to find the optimal cost function by going over a number of iterations.
Visualization of the squared error (from Setosa.io)
The equation for this model is referred to as the cost function and is a way to find the optimal error by minimizing and measuring it.
The gradient descent algorithm is used to find the optimal cost function by going over a number of iterations.
The gradient descent algorithm is used to find the optimal cost function by going over a number of iterations.
We will tune the model by iterating over a number of alpha and lambda pairs and we can see which pair has the lowest associated error.
The gradient descent algorithm is used to find the optimal cost function by going over a number of iterations.
We will tune the model by iterating over a number of alpha and lambda pairs and we can see which pair has the lowest associated error.
Equation for least ordinary squares
One situation is the data showing multi-collinearity, this is when predictor variables are correlated to each other and to the response variable.
We can see that the R mean-squared values using all three models were very close to each other, but both did marginally perform better than ridge regression (Lasso having done best).
To produce a more accurate model of complex data we can add a penalty term to the OLS equation.
These are known as L1 regularization(Lasso regression) and L2 regularization(ridge regression).The best model we can hope to come up with minimizes both the bias and the variance:

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.