

## **LAB ASSIGNMENT-2**

---

KARTHIK GUNDALAPALLI

SANTOSH GANDHI DIDI

MOURYA PRAHARSHA BOBBILI

SAI TEJA PENUGONDA

(1Q) Pick any dataset from the dataset sheet in the class sheet or online which includes both numeric and non-numeric features

a. Perform exploratory data analysis on the data set (like Handling null values, removing the features not correlated to the target class, encoding the categorical features, ...)

b. Apply the three classification algorithms Naïve Baye's, SVM and KNN on the chosen data set and report which classifier gives better result.

Approach: we have applied Naive bayes ,SVM and KNN on the train dataset in .csv format and the results are displayed below:

Code with output:

The screenshot shows a terminal window and an IPython console side-by-side. The terminal window displays a Python script named lab2-4.py. The script imports pandas, sklearn, and warnings, reads a CSV file, fills missing values with mean, drops columns, encodes categorical data, and performs SVM classification. The IPython console shows the data summary (shape: 891x8), feature types, and classification results (precision, recall, f1-score, support for SVM and KNN).

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#
# Created on Wed Mar 13 20:08:56 2019
# @author: karthikchowdary
#
# Import pandas as pd
from sklearn import datasets
from sklearn.model_selection import train_test_split
import warnings
warnings.simplefilter("ignore")
train = pd.read_csv('/Users/karthikchowdary/Desktop/Karthik/graduate/spring-19/python/lab-2/titanic.csv')
train.isna().head()
train.fillna(train.mean(), inplace=True)
train.info(),
train = train.drop(['Name', 'Ticket', 'Cabin', 'Embarked'], axis=1)
# Encoding categorical data
from sklearn.preprocessing import LabelEncoder
from sklearn import preprocessing
labelEncoder = preprocessing.LabelEncoder()
labelEncoder.fit(train['Sex'])
train['Sex'] = labelEncoder.transform(train['Sex'])

train.info()
# calculating the svm
feature_cols = ['PassengerId', 'Pclass', 'Age', 'SibSp', 'Parch', 'Fare', 'Sex']
from sklearn.svm import SVC
X = train[feature_cols]
y = train.Survived
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
svm = SVC()
expected = train.Survived
svm.fit(X_train, y_train)

[[465  84]
 [100 242]]
accuracy using Gaussian navie bayes Model is 79.3296089385475
accuracy using knn Model is 0.75308864197530864
```

In [21]:

	precision	recall	f1-score	support
0	0.87	0.99	0.92	549
1	0.98	0.75	0.85	342
micro avg	0.90	0.90	0.90	891
macro avg	0.92	0.87	0.89	891
weighted avg	0.91	0.90	0.90	891

	precision	recall	f1-score	support
0	0.82	0.85	0.83	549
1	0.74	0.71	0.72	342
micro avg	0.79	0.79	0.79	891
macro avg	0.78	0.78	0.78	891
weighted avg	0.79	0.79	0.79	891

```

lab2-1.py* lab2-2.py lab2-3.py lab2-4.py
Editor - /Users/karthikchowdary/Desktop/Karthik/graduate/spring-19/python/lab-2/
File Edit View Insert Cell Kernel Help
lab2-1.py
52 svm = SVC()
53 expected = train.Survived
54
55 svm.fit(X_train, y_train)
56
57 predicted_label = svm.predict(train[feature_cols])
58
59
60 # Cross Validation compare the predicted and expected values
61 print(metrics.confusion_matrix(expected, predicted_label))
62
63 # Accuracy of SVM classifier : 0.81
64 print(metrics.classification_report(expected, predicted_label))
65
66
67 from sklearn.naive_bayes import GaussianNB
68 model = GaussianNB()
69 model.fit(train[feature_cols], train.Survived)
70
71 expected = train.Survived
72 predicted = model.predict(train[feature_cols])
73
74
75 print(metrics.classification_report(expected, predicted))
76 print(metrics.confusion_matrix(expected, predicted))
77 X_train, X_test, Y_train, Y_test = train_test_split(train[feature_cols], train.Survived, test_size=0.2, random_state=42)
78
79 model.fit(X_train, Y_train)
80
81 Y_predicted = model.predict(X_test)
82
83 print("accuracy using Gaussian navie bayes Model is ", metrics.accuracy_score(Y_test, Y_predicted))
84
85
86 from sklearn.linear_model import LogisticRegression
87 logreg = LogisticRegression()
88 logreg.fit(X, y)
89 logreg.predict(X)
90 logreg.predict(X)
91 y_pred = logreg.predict(X)
92
93
94 from sklearn.neighbors import KNeighborsClassifier
95 knn = KNeighborsClassifier(n_neighbors=5)
96 knn.fit(X, y)
97 y_pred = knn.predict(X)
98
99 print("accuracy using knn Model is ",metrics.accuracy_score(y, y_pred))
100
101
[[465 84]
 [100 242]]
accuracy using Gaussian navie bayes Model is 79.3296089385475
accuracy using knn Model is 0.7530864197530864
In [21]:

```

Permissions: RW End-of-lines: LF Encoding: UTF-8 Line: 102 Column: 12 Memory: 62 %

(2Q) Choose any dataset of your choice. Apply K-means on the dataset and visualize the clusters using matplotlib or seaborn.

a. Report which K is the best using the elbow method.

b. Evaluate with silhouette score or other scores relevant for unsupervised approaches  
(before applying clustering clean the data set with the EDA learned in the class)

Approach: we applied k-means on the stock-data and the output is visualized

Code and output:

The screenshot shows a Jupyter Notebook environment with several files listed in the left sidebar: lab2-1.py, lab2-2.py, 2.py, lab2-3.py, and lab2-4.py. The main area displays a Python script for K-Means clustering and a corresponding scatter plot.

**Python Script Content:**

```
1 from sklearn.cluster import KMeans
2 from sklearn.model_selection import train_test_split
3 import pandas as pd
4 import numpy as np
5 from sklearn.metrics import silhouette_score
6 import matplotlib.pyplot as plt
7 data=pd.read_csv('sample_stocks.csv')
8 data.dropna(inplace=True)
9 data['date'].dt.year
10 train, test = train_test_split(data, test_size=0.2)
11 clf = KMeans(n_clusters=3)
12 answer=clf.fit_predict(data)
13 print(score)
14 color_theme=np.array(['darkgray','lightsalmon','powderblue'])
15 plt.scatter(x=data['returns'],y=data['dividendyield'],c=color_theme[clf.labels_],s=50)
16 plt.show()
17 #print(prediction)
```

**Output Console:**

```
In [37]: runfile('/Users/karthikchowdary/Desktop/Karthik/graduate/spring-19/python/lab-2/lab2-2.py', wdir='/Users/karthikchowdary/Desktop/Karthik/graduate/spring-19/python/lab-2')
0.7911020747156259
```

**Scatter Plot:**

A scatter plot showing data points categorized into three clusters based on their 'returns' and 'dividendyield'. The x-axis ranges from -20 to 40, and the y-axis ranges from 0 to 5. The clusters are represented by different colors: dark gray, light salmon, and powder blue.

Editor - /Users/karthikchowdary/Desktop/Karthik/graduate/spring-19/python/lab-2

lab2-1.py lab2-2.py 2.py lab2-3.py lab2-4.py | IPython console

```

1 from sklearn.cluster import KMeans
2 from sklearn.model_selection import train_test_split
3 import pandas as pd
4 import matplotlib.pyplot as plt
5 data=pd.read_csv('sample_stocks.csv')
6 #train,test=train_test_split(data,test_size=0.2)
7 #X,y=load_squared_err()
8 K=krange(1,15)
9 for K in K:
10     clf = KMeans(n_clusters=k)
11     clf.fit(data)
12     Sum_of_Squared_err.append(clf.inertia_)
13 plt.plot(K,Sum_of_Squared_err,'bx-')
14 plt.xlabel("K values")
15 plt.ylabel("sum_of_squared_error")
16 plt.show()
17
18 #prediction=clf.predict(test)
19 #print(prediction)

```

In [37]: runfile('/Users/karthikchowdary/Desktop/Karthik/graduate/spring-19/python/lab-2/lab2-2.py', wdir='/Users/karthikchowdary/Desktop/Karthik/graduate/spring-19/python/lab-2')
0.7911028747156259

In [38]: runfile('/Users/karthikchowdary/Desktop/Karthik/graduate/spring-19/python/lab-2/2.py', wdir='/Users/karthikchowdary/Desktop/Karthik/graduate/spring-19/python/lab-2')

In [39]: |

Run file Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 1 Column: 1 Memory: 66 %

(3) Write a program in which take an Input file, use the simple approach below to summarize

a text file:Link to input file: <https://umkc.box.com/s/7by0f4540cdbdp3pm60h5ffefsvrwa>.

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

Approach: All the techniques are executed in the program and can be seen in following screen shots

Code and Output:

The screenshot shows the Spyder Python 3.7 IDE interface. On the left, the code editor displays a script named 'test.py' with the following content:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sat Feb 23 15:41:57 2019
4
5 @author: santosh
6 """
7 from bs4 import BeautifulSoup
8 from bs4.element import Comment
9 import urllib.request
10 from nltk.stem import PorterStemmer
11 from nltk.stem import WordNetLemmatizer
12 from nltk import wordpunct_tokenize, pos_tag, ne_chunk, ngrams, FreqDist, sent_tokenize, word_tokenize
13
14
15
16 file=open("nlp_input.txt","r+")
17 fileData = file.read()
18
19 # Word Tokenization - to extract each word from the text
20 tokens = word_tokenize(fileData)
21
22 # Applying Lemmatization
23 lemmatizer = WordNetLemmatizer()
24 lemmatizerOutput = []
25 print("Lemmatization Output : \n")
26 for tok in tokens:
27     # Iterating through each word and Lemmatizing and appending it to list
28     lemmatizerOutput.append(lemmatizer.lemmatize(str(tok)))
29 print(lemmatizerOutput)
30
31 # performing trigram on the Lemmatizer Output
32 print("trigrams :\n")
33 trigramsOutput = []
34 for tri in ngrams(tokens, 3):
35     # Fetching trigrams using 'ngrams' method and Iterating it and appending it to list
36     trigramsOutput.append(tri)
37 print(trigramsOutput)
38
39 # triGram-Word Frequency
40 # Using trigramsOutput fetch the WordFreq Details
41 wordFreq = FreqDist(trigramsOutput)
42 # Getting Most Common words and Printing them - Will get the Counts from top to least
mostCommon = wordFreq.most_common()
```

The right side of the interface shows the 'Source Console' and 'IPython console'. The 'Source Console' pane displays the 'Usage' help information for the 'ctrl+1' key. The 'IPython console' pane shows the output of the code execution, including the lemmatized words and the resulting trigrams. The bottom status bar indicates the current permissions, encoding, line number, column number, memory usage, and the date and time (10:04 PM, 3/13/2019).

The screenshot shows the Spyder Python IDE interface. The main window displays a Python script named `test.py` with the following code:

```

24 lemmatizerOutput = []
25 print("Lemmatization Output : \n")
26 for tok in tokens:
27     # Iterating through each word and Lemmatizing and appending it to list
28     lemmatizerOutput.append(lemmatizer.lemmatize(str(tok)))
29 print(lemmatizerOutput)
30
31 # performing trigram on the Lemmatizer Output
32 print("trigrams :")
33 trigramsOutput = []
34 for tri in ngrams(tokens, 3):
35     # Fetching trigrams using 'ngrams' method and Iterating it and appending it to list
36     trigramsOutput.append(tri)
37 print(trigramsOutput)
38
39 # triGram- Word Frequency
40 # Using trigramsOutput fetch the WordFreq Details
41 wordFreq = FreqDist(trigramsOutput)
42 # Getting Most Common Words and Printing them - Will get the Counts from top to least
43 mostCommon = wordFreq.most_common()
44 print("trigrams Frequency (From Top to Least) : \n", mostCommon)
45 # Fetching the Top 10 trigrams
46 top10 = wordFreq.most_common(10)
47 print("Top 10 trigrams : \n", top10)
48
49
50 # Getting Sentences using Sentence Tokenization
51 sentTokens = sent_tokenize(fileData)
52 # Creating an Array to append the sentence
53 concatenatedArray = []
54 # Iterating the Sentences
55 for sentence in sentTokens:
56     # Iterating the trigrams present
57     for a, b, c in trigramsOutput:
58         # Iterating the Top 10 trigrams
59         for ((d, e, f), length) in top10:
60             # Comparing the each with each of the Top 10 trigram
61             if(a == d and e == f):
62                 concatenatedArray.append(sentence)
63
64 print("Concatenated Array : ", concatenatedArray)
65 print("Maximum of Concatenated Array : ", max(concatenatedArray))

```

To the right of the code editor, there is a "Usage" help panel with the following text:

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.  
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our [tutorial](#)

The bottom right corner of the interface shows the system tray with icons for battery, signal, and date/time.

(4) Create Multiple Regression by choosing a dataset of your choice (again before evaluating, clean the data set with the EDA learned in the class). Evaluate the model using RMSE and R2 and also report if you saw any improvement before and after the EDA.

Approach: We applied multiple regression on the diabetes data set and also evaluated using RMSE and R2 and the results can be seen as follows.

Code and Output:

The screenshot shows a Jupyter Notebook interface with several tabs at the top: lab2-1.py, lab2-2.py, 2.py, lab2-3.py, and lab2-4.py. The lab2-4.py tab is active, displaying Python code for data cleaning, EDA, and model training. The code includes imports for numpy, matplotlib.pyplot, pandas, and warnings, along with handling missing values and grouping data. It then uses pd.get\_dummies to handle categorical variables like 'Pregnancies' and 'DiabetesPedigreeFunction'. The notebook then splits the data into X\_train, X\_test, y\_train, and y\_test using train\_test\_split with a test\_size of 0.25. A LinearRegression model is fitted to the training data, and predictions are made for the test set. Finally, it calculates the mean squared error and prints the results.

```

1 #!/usr/bin/env python3
2 #-- coding: utf-8 --
3 #
4 Created on Wed Mar 13 21:33:05 2019
5
6 @author: karthikchowdary
7
8
9 # Multiple Linear Regression
10
11 # Importing the libraries
12 import numpy as np
13 import matplotlib.pyplot as plt
14 import warnings
15 warnings.simplefilter("ignore")
16 import pandas as pd
17 import warnings
18 warnings.simplefilter("ignore")
19 # Importing the dataset
20 dataset = pd.read_csv('/Users/karthikchowdary/Desktop/Karthik/graduate/spring-19/python/lab-2/diabetes.csv')
21
22 dataset.describe()
23 dataset['Insulin'].value_counts()
24 dataset.groupby(['Insulin', 'BMI']).mean()
25
26 dataset = dataset.fillna(dataset.mean())
27
28
29 #X = dataset.iloc[:, :-1].values
30 y = dataset.iloc[:, 2].values
31 X = dataset.drop(['Pregnancies', 'Insulin', 'SkinThickness', 'BMI'], axis=1)
32 #dataset = dataset.fillna(dataset.mean())
33
34
35
36
37 #df = df_train.drop(['Summary', 'Daily Summary'], axis=1)
38
39 #X = pd.get_dummies(X, columns=['Precip Type'])
40 #before EDA
41 from sklearn.model_selection import train_test_split
42 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state=0)
43 from sklearn.linear_model import LinearRegression
44 regressor = LinearRegression()
45 model = regressor.fit(X_train, y_train)
46
47 # Predicting the Test set results
48 y_pred = regressor.predict(X_test)
49
50 from sklearn.metrics import mean_squared_error, r2_score
51 print("Variance score: %2f" % r2_score(y_test,y_pred))
52 print("Mean squared error: %2f" % mean_squared_error(y_test,y_pred))
53
54 # Encoding categorical data
55 from sklearn.preprocessing import LabelEncoder, OneHotEncoder
56 labelencoder = LabelEncoder()
57 X.values[:, 1] = labelencoder.fit_transform(X.values[:, 1])

```

The right side of the interface shows two plots. The first plot is a scatter plot of 'SkinThickness' vs 'Insulin' with points colored by 'DiabetesPedigreeFunction'. The second plot is a line graph showing the sum of squared error versus the number of K values, which decreases rapidly from approximately 250,000 to near zero as K increases from 2 to 14.

In [38]: runfile('/Users/karthikchowdary/Desktop/Karthik/graduate/spring-19/python/lab-2/2.py', wdir='/Users/karthikchowdary/Desktop/Karthik/graduate/spring-19/python/lab-2')

In [39]: runfile('/Users/karthikchowdary/Desktop/Karthik/graduate/spring-19/python/lab-2/lab-4.py', wdir='/Users/karthikchowdary/Desktop/Karthik/graduate/spring-19/python/lab-2')

Variance score: 1.00  
Mean squared error: 0.00  
Insulin 1.000000  
SkinThickness 0.436783  
Glucose 0.331357  
Name: Insulin, dtype: float64  
BloodPressure 0.088933  
Age -0.042163  
Pregnancies -0.073535  
Name: Insulin, dtype: float64  
Variance score: 1.00  
Mean squared error: 0.00

In [40]: