PYTHON DEEP LEARNING

PROJECT_BASED_EXAM-1 REPORT

TEAM MEMBERS:

ARIKATLA, PALLAVI – 4

GENTE, HARUN SAI KUMAR – 15

KOTA, SARIKA REDDY – 25

MAROJU, JAGADEESH – 28

INTRODUCTION:

Primary moto of this project is to implement Deep Learning algorithms in Python. This project majorly uses KNN, Naïve bayes, SVM and elbow methods.

Software Required:

- Jupyter notebook (with installed plotting libraries)
- Python 3.

OBJECTIVE:

To implement different algorithms and models learnt in Python Deep Learning so far. And to identify better model by calculating score and model efficiencies.

METHODS:

Different algorithms and methods used:

- KNN – K nearest neighbors is a modest algorithm that aids in categorizing cases based on resemblance.
- Naïve bayes – Classification of two or multiple classes with the help of bayes theorem.
- SVM – Support Vector Machines are usually used for classification, regression and detection and removal of outliers.
- Elbow method – It helps running Kmeans clustering and used to determine number of clusters in the given dataset.

WORK FLOW:

QUESTION 1:

a)Apply Any classification of your choice (KNN, Naïve Bayes, SVM, Random Forest, ...) and report the performance.

Procedure:

1. Loaded the data using pandas library and created data frame.

2. As the target column is "class", sliced the dataset accordingly target column into y_train and all other columns into x_train.

3. Split the data into test and train using train_test_spilt with the probability of test size 0.4 and with the random_state=0.

4. Created GaussianNB() object to implement Naive Bayes algorithm.

5. Found predictions using X_Test data.

6. Evaluated the model by finding the accuracy score for the test data.

7. Got the classification report for the test and predicated data.

8. Applied KNN classification algorithm.

9. Now calculated the score for the test data.

CODE:

```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.neighbors import KNeighborsClassifier
import pandas as pd


credit_card_df = pd.read_csv('./creditcard.csv')
X_train = credit_card_df.drop("Class",axis=1)
Y_train = credit_card_df["Class"]
X_train, X_test, Y_train, Y_test= train_test_split(X_train, Y_train, test_size=0.4, random_state=0)
model = GaussianNB()
model.fit(X_train,Y_train)
y_pred = model.predict(X_test)
score = accuracy_score(Y_test,y_pred)*100
print("accuracy score: " + str(score))

print(classification_report(Y_test, y_pred))

#KNN
knn = KNeighborsClassifier(n_neighbors = 5)
knn.fit(X_train,Y_train)
score = knn.score(X_test,Y_test)*100
print("KNN socre: " + str(score))
```

OUTPUT:

```
accuracy score: 99.32059373436444
              precision    recall  f1-score   support

           0       1.00      0.99      1.00    113724
           1       0.15      0.63      0.25       199

    accuracy                           0.99    113923
   macro avg       0.58      0.81      0.62    113923
weighted avg       1.00      0.99      1.00    113923

KNN socre: 99.8323428982734
```

b) Visualize the number of samples per class.

Procedure:

1. Used matplot library to visualize the required data.

2. Found fraud and non-fraud transactions from the data set.

3. Drawn area plot to visualize the fraud and non-fraud data.

CODE:

```
# ##visualization
# import matplotlib.pyplot as plt
credit_card_df["Class"].value_counts()
# print(credit_card_df["Class"].value_counts())
# plt.plot(credit_card_df["Class"].value_counts())
# credit_card_df.Class
credit_card_df["Class"].value_counts().plot(kind='area')
# credit_card_df["Class"].value_counts().nlargest(40).plot(kind='bar',figsize=(10,5))
```

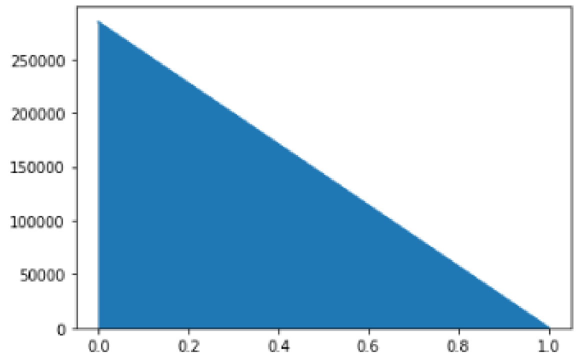
OUTPUT:

Samples:

```
In [3]: # ##visualization
        # import matplotlib.pyplot as plt
        credit_card_df["Class"].value_counts()
        print(credit_card_df["Class"].value_counts())
        # plt.plot(credit_card_df["Class"].value_counts())
        # credit_card_df.Class
        #credit_card_df["Class"].value_counts().plot(kind='area')
        # credit_card_df["Class"].value_counts().nlargest(40).plot(kind='bar',figsize=(10,5))

        0    284315
        1       492
        Name: Class, dtype: int64
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x26c83bef9c8>
```



c) Discuss challenges faced while dealing with imbalanced datasheet.

<u>Problems with imbalanced datasheet:</u>

1. The main problem while dealing with imbalanced dataset is that ML algorithms produce wrong results. This is because these algorithms show bias towards the majority class.

2. ML algorithms won't consider minority class since they are very less in dataset.

3. Consider we have 1% minority class and 99% majority class data then ML algorithms treats all of them belongs to majority class.

<u>Handling imbalanced datasheet:</u>

There are mainly two methods to handle this problem,

1. <u>Oversampling:</u> This method removes the imbalance in the datasheet by creating new minority class instances.

We have 4 types of methods in Over sampling

i. <u>Random oversampling:</u>

This method adds the minority class instances randomly by replicating the existing minority class instances. It results in over fitting.

ii. <u>Cluster oversampling:</u>

In this method, KMeans algorithm will be applied to find the cluster. After that each cluster will be designed to have equal number of instances.

We will have over fitting in this method too.

iii. <u>Synthetic oversampling</u>: In this method, a small portion of minority class will be selected and these subsets are created to balance the data.

This will eliminate over fitting.

iv. Modified Synthetic Oversampling: This is same as synthetic oversampling but inherits distribution of the minority classes.

2. Under sampling:

In this method, the imbalance will be reduced by concentrating on the majority class. To do this we have a method "Random undersampling"

In Random undersampling, existing majority classes will be eliminated randomly. This method is not safe because it even eliminates the useful data.

QUESTION 2:

a) Apply K-means on the data set, report K using elbow method.

1. Import the necessary libraries from scikit library.

2. create the data frame using the dataset

3. To find null values in the columns use isnull() method.

4. using input data and plot the graph with elbow method then found the number of clusters.

5. Here we got 5 clusters.

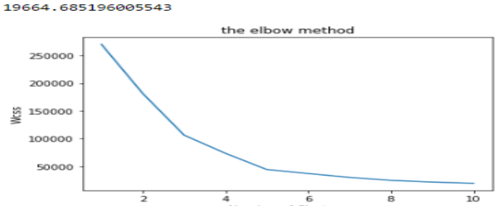
6. Using the below plot, we can infer that their optimized k value is 5.

CODE:

```
x = data_frame.iloc[:, [3,4]].values
y = data_frame.iloc[:-1]
wcss= []
for i in range(1,11):
    kmeans= KMeans(n_clusters=i,max_iter=300,random_state=0)
    kmeans.fit(x)
    print(kmeans.inertia_)
    wcss.append(kmeans.inertia_)
plt.plot(range(1,11),wcss)
plt.title('the elbow method')
plt.xlabel('Number of Clusters')
plt.ylabel('Wcss')
plt.show()
```

OUTPUT:

```
269981.28
181363.59595959596
106348.37306211118
73679.78903948834
44448.45544793371
37239.83554245604
30273.394312070042
25018.576334776335
21850.165282585633
19664.685196005543
```



b) Evaluate with silhouette score or other scores relevant for unsupervised approaches.

<u>Procedure:</u>

1. create KMeans Algorithm model.
2. Now fit x data to into the model.
3. Found prediction values passing x values.
4. Calculate score actual and predicted.

CODE:

```
In [3]: k = KMeans(5)
        k.fit(x)
        preditction = k.predict(x)
        score = metrics.silhouette_score(x, preditction)
        print(score)
```

OUTPUT:

```
0.553931997444648
```

c) Visualize cluster result.

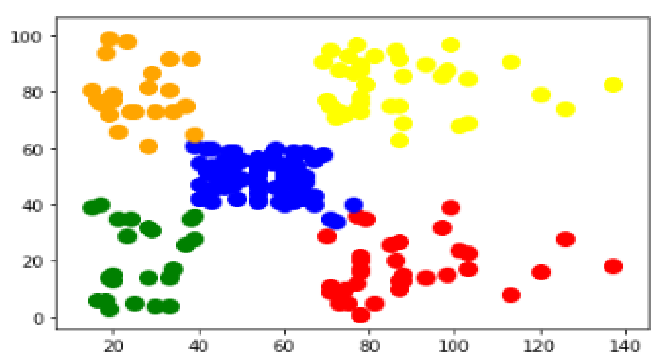
1. Used matplot library and drawn a scatter plot for the clusters.

CODE:

```
plt.scatter(x[preditction == 0, 0], x[preditction == 0, 1], s = 100, c = 'red')
plt.scatter(x[preditction == 1, 0], x[preditction == 1, 1], s = 100, c = 'blue')
plt.scatter(x[preditction == 2, 0], x[preditction == 2, 1], s = 100, c = 'green')
plt.scatter(x[preditction == 3, 0], x[preditction == 3, 1], s = 100, c = 'yellow')
plt.scatter(x[preditction == 4, 0], x[preditction == 4, 1], s = 100, c = 'orange')
```

OUTPUT:

```
<matplotlib.collections.PathCollection at 0x1fe38bd19c8>
```



QUESTION 3:

a) Apply some Exploratory Data Analysis on the given data set to draw some insight from the data.

- Eliminate unnecessary columns:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

data_frame = pd.read_csv('./weather.csv')

##removing unnecessary columns
data_frame = data_frame.drop(['Formatted Date','Precip Type','Summary','Daily Summary'],axis=1)
data_frame.head(5)
```

	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)
0	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13
1	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63
2	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94
3	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41
4	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51

- Eliminate duplicate columns:

```
#drop duplicate columns
duplicate_rows = data_frame[data_frame.duplicated()]
print(duplicate_rows.shape)
data_frame = data_frame.drop_duplicates()
```

```
(0, 8)
```

- Remove null values:

```
#remove null values
print(data_frame.isnull().sum())
data_frame = data_frame.dropna()
print(data_frame.isnull().sum())
```

```
Temperature (C)             0
Apparent Temperature (C)    0
Humidity                    0
Wind Speed (km/h)           0
Wind Bearing (degrees)      0
Visibility (km)             0
Loud Cover                  0
Pressure (millibars)        0
dtype: int64
Temperature (C)             0
Apparent Temperature (C)    0
Humidity                    0
Wind Speed (km/h)           0
Wind Bearing (degrees)      0
Visibility (km)             0
Loud Cover                  0
Pressure (millibars)        0
dtype: int64
```

b) Visualize the data and draw the model line.

1. Imported the necessary libraries.

2. Red the weather.csv dataset and create a dataframe.

3. Remove the unnecessary columns from the dataframe using drop.

4. Now drop the duplicate columns.

5. Removed the Null values using the isnull() method.

6. using seaborn library, visualized the temperature varaince and humidity variance.

7. Now drawn a scatter plot for Temperature vs Windspeed and found the outliers.

8. Removed the outliers found from the above plot and drawn the new scatter plot.

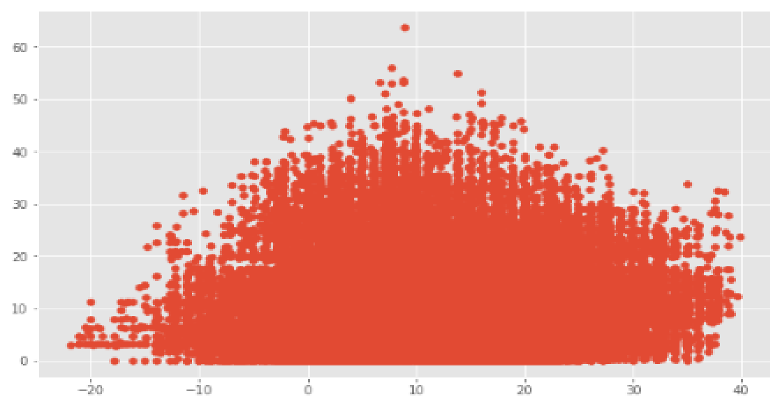
9. Drawn plot for temperature vs Visibility and found outliers.

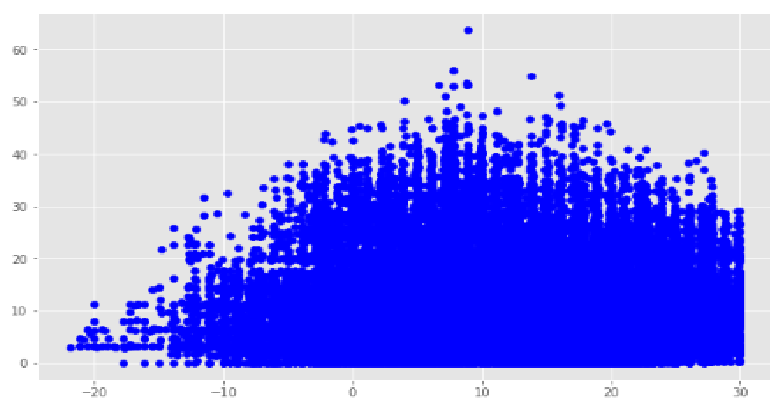
10. Removed the outliers found from the above plot and drawn the new plot.

CODE:

```
#2nd plot
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt1
fig, ax = plt.subplots(figsize=(10,6))
plt.scatter(data_frame['Temperature (C)'],data_frame['Wind Speed (km/h)'])
plt.show()
filter_data = data_frame[(data_frame['Temperature (C)'] < 40)]
plt1.scatter(filter_data['Temperature (C)'],filter_data['Wind Speed (km/h)'],color='b')
plt1.show()
```

OUTPUT:





c) Evaluate the model and try to interpret the performance that you get.

1. Import pandas, Numpy and also Train_test_spilt and Linear model from scikit library
2. Get the numeric_features from the data frame.
3. Find the top five positive correlated values by taking temperature column
4. To remove null values by using isnull() method
5. Here we got all the Null values as zero.
6. Now Handling the missing values using interpolate() method
7. Split the data as train and test data using train_test_split.
8. Create the model for linear regression.
9. Trained the model with the train data.
10. Calculated R2 score and RMSE score.

CODE:

```
##Regression model
from sklearn.preprocessing import LabelEncoder


n_features = data_frame.select_dtypes(include=[np.number])


#delete null values
nulls = pd.DataFrame(data_frame.isnull().sum().sort_values(ascending=False))
nulls.columns = ['Null Count']
nulls.index.name = 'Feature'
print(nulls)
print("\n")


# ##handling missing value
data_frame = data_frame.select_dtypes(include=[np.number]).interpolate().dropna()
print("missing values: " + str(sum(data_frame.isnull().sum() != 0))+ "\n")


X = data_frame.drop(["Temperature (C)"],axis=1)
y = data_frame["Temperature (C)"]


X_train, X_test, y_train, y_test = train_test_split(
                                    X, y, random_state=42, test_size=0.4)


l_model = linear_model.LinearRegression()
model = l_model.fit(X_train, y_train)
print ("R2 score: ", model.score(X_test, y_test))


prdc = model.predict(X_test)
print ('RMSE score: ', mean_squared_error(y_test, prdc))
```

OUTPUT:

```
                              Null Count
Feature
Pressure (millibars)                   0
Loud Cover                             0
Visibility (km)                        0
Wind Bearing (degrees)                 0
Wind Speed (km/h)                      0
Humidity                               0
Apparent Temperature (C)               0
Temperature (C)                        0


missing values: 0

R2 score:  0.9901112877065714
RMSE score:  0.9033461263374988
```

As the R2 score is 0.99, we can say that the model is very close to the fitted line.

QUESTION 4:

Use the given dataset and apply different classifications.

<u>Procedure:</u>

1. First import the necessary libraries.

2. Read the spam.csv file as we cannot read it directly, we have to encode it.

3. Now clean the text data by dropping the unnecessary columns.

```
import pandas as pd


spam_data = pd.read_csv('./spam.csv',encoding = "latin-1", engine='python')
spam_data = spam_data.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'],axis=1)
spam_data.head(5)
spam_data.describe()
spam_data.groupby('Class').describe()
```

	Text			
	count	unique	top	freq
Class				
ham	4825	4516	Sorry, I'll call later	30
spam	747	653	Please call our customer service representativ...	4

4. Initialized the countvectorizer.

5. Found out the shape of Text data of the given dataset.

```
from sklearn.feature_extraction.text import CountVectorizer

bow_transformer = CountVectorizer()

bow_transformer.fit(spam_data['Text'])
```

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                lowercase=True, max_df=1.0, max_features=None, min_df=1,
                ngram_range=(1, 1), preprocessor=None, stop_words=None,
                strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None, vocabulary=None)
```

6. Initialized the Tfidf transformer.

7. Using Tfidf found out each word idf_weight score (more used word means less score) of text column.

```
from sklearn.feature_extraction.text import TfidfTransformer

tfidf_transformer=TfidfTransformer(smooth_idf=True,use_idf=True)
tfidf_transformer.fit(word_count_vector)

df_idf = pd.DataFrame(tfidf_transformer.idf_, index=cv.get_feature_names(),columns=["idf_weights"])

df_idf.sort_values(by=['idf_weights'])
```

	idf_weights
to	2.198545
you	2.254829
the	2.689346
in	2.933605
and	2.947347
...	...
bleh	8.932542
mee	8.932542
blimey	8.932542
mirror	8.932542
ûówell	8.932542

8672 rows × 1 columns

8. Now loop throw each text data column for 3 rows and found tfidf score.

CODE:

```
count_vector=cv.transform(spam_data.Text)
tf_idf_vector=tfidf_transformer.transform(count_vector)

feature_names = cv.get_feature_names()
i = 0
for x in tf_idf_vector:
    i+= 1
    df = pd.DataFrame(x.T.todense(), index=feature_names, columns=["tfidf"])
    print(df.sort_values(by=["tfidf"],ascending=False))
    if i == 3:
        break
```

OUTPUT:

```
              tfidf
jurong       0.326425
amore        0.326425
buffet       0.311608
bugis        0.275765
cine         0.275765
...               ...
electricity  0.000000
elections    0.000000
election     0.000000
eldest       0.000000
ûówell       0.000000

[8672 rows x 1 columns]
           tfidf
oni       0.546588
joking    0.523646
wif       0.431601
lar       0.408299
ok        0.272120
...            ...
election  0.000000
eldest    0.000000
elaya     0.000000
elama     0.000000
ûówell    0.000000

[8672 rows x 1 columns]
                     tfidf
fa                 0.460253
entry              0.352710
08452810075over18  0.230126
2005               0.222362
21st               0.222362
...                     ...
electricity        0.000000
elections          0.000000
election           0.000000
eldest             0.000000
ûówell             0.000000

[8672 rows x 1 columns]
```

9. Applied CountVectorizer,TfidfTransformer and MultinomialNB.

10. Trained the model and found prediction.

11. Got classification report using prediction and test data.

```python
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report,confusion_matrix


# messages = pd.read_csv('./spam.csv', encoding='latin-1')
# messages.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'],axis=1,inplace=True)
# messages = messages.rename(columns={'v1': 'Class','v2': 'Text'})

x_train, x_test, y_train, y_test = train_test_split(spam_data['Text'],spam_data['Class'],test_size=0.2)

model = Pipeline([
    ('bow',CountVectorizer()),
    ('tfidf',TfidfTransformer()),
    ('classifier',MultinomialNB())
])

model.fit(x_train,y_train)

prediction = pipeline.predict(x_test)

print(classification_report(y_test,prediction))
```

```
              precision    recall  f1-score   support

         ham       0.96      1.00      0.98       961
        spam       1.00      0.77      0.87       154

    accuracy                           0.97      1115
   macro avg       0.98      0.88      0.92      1115
weighted avg       0.97      0.97      0.97      1115
```

QUESTION 5:

a)Pick any dataset online for theclassification problem which includes both numeric and non-numeric features and Perform exploratory data analysis.

Procedure:

1. Import necessary libraries.

2. we have picked train.csv dataset as it has both numeric and non-numeric features.

3. Now using data_frame.info, printed all the columns

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

data_frame = pd.read_csv('./train.csv')

data_frame.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   int64
 2   MSZoning       1460 non-null   object
 3   LotFrontage    1201 non-null   float64
 4   LotArea        1460 non-null   int64
 5   Street         1460 non-null   object
 6   Alley          91 non-null     object
 7   LotShape       1460 non-null   object
 8   LandContour    1460 non-null   object
 9   Utilities      1460 non-null   object
 10  LotConfig      1460 non-null   object
 11  LandSlope      1460 non-null   object
 12  Neighborhood   1460 non-null   object
 13  Condition1     1460 non-null   object
 14  Condition2     1460 non-null   object
 15  BldgType       1460 non-null   object
 16  HouseStyle     1460 non-null   object
 17  OverallQual    1460 non-null   int64
 18  OverallCond    1460 non-null   int64
 19  YearBuilt      1460 non-null   int64
 20  YearRemodAdd   1460 non-null   int64
 21  RoofStyle      1460 non-null   object
 22  RoofMatl       1460 non-null   object
 23  Exterior1st    1460 non-null   object
 24  Exterior2nd    1460 non-null   object
```

4. Found out the unique features in Lotshape and also nulls using isnull() method.

5. we observed that there are no nulls.

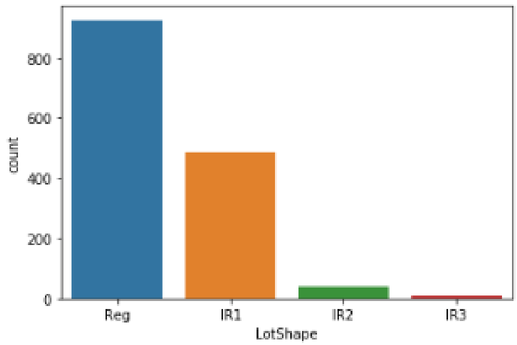
```
print(data_frame['LotShape'].unique())
print(data_frame['LotShape'].isnull().sum())
```

```
['Reg' 'IR1' 'IR2' 'IR3']
0
```

6. plotting the frequency of features in Lotshape

```
#frequency of LotShape
sns.countplot(data = data_frame, x = 'LotShape')
```

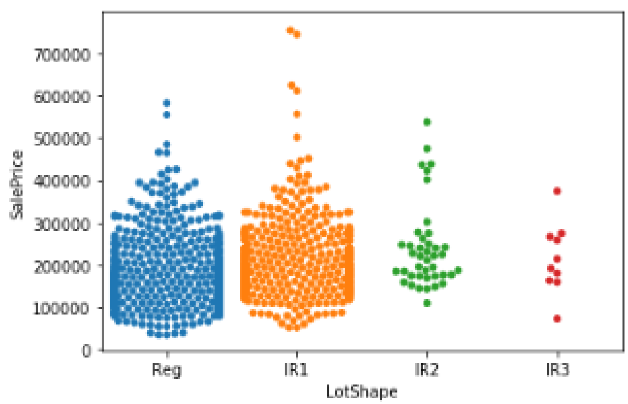
<matplotlib.axes._subplots.AxesSubplot at 0x11b161a1208>



7. Drawn Swarmplot between Lotshape and frequency.

```
#second plot
sns.swarmplot(data = data_frame, x='LotShape', y='SalePrice')
```

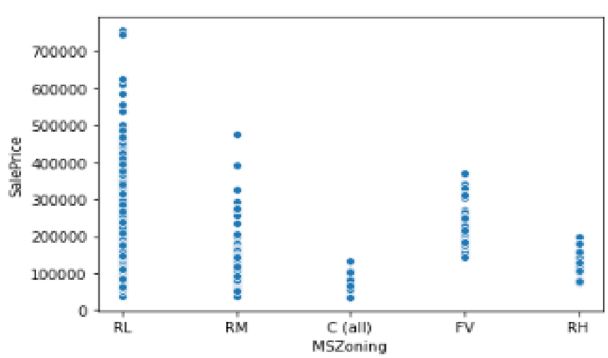
<matplotlib.axes._subplots.AxesSubplot at 0x11b1853a508>



8. Drawn Scatterplot between MSZoning and saleprice.

```
sns.scatterplot(data = data_frame, x='MSZoning', y='SalePrice')
```

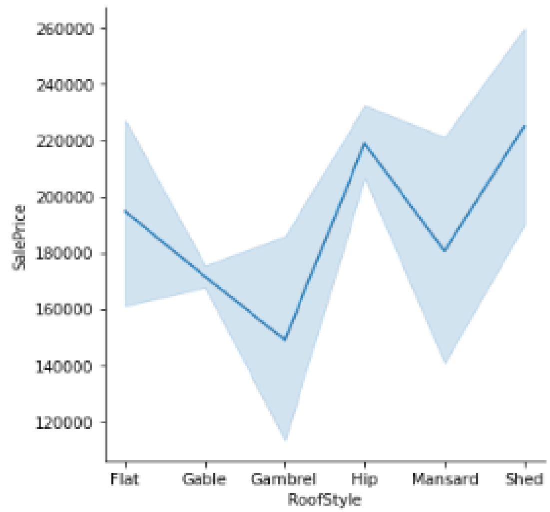
<matplotlib.axes._subplots.AxesSubplot at 0x11b185d5a88>



9. Drawn Relplot between Roofstyle an saleprice.

```
sns.relplot(data = data_frame, x='RoofStyle', y='SalePrice',kind='line')
```

```
<seaborn.axisgrid.FacetGrid at 0x11b18658c88>
```



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

<u>Procedure:</u>

1. Import all the necessary libraries from scikit library.

2. now check for any Null values and remove them.

3. Slicing the dataframe with all the columns except target column i.e, saleprice in X and saleprice in y.

4. Spilt the dataset using train_test_split into train and test.

5. Now calculating score using Naive bayes classification algorithm.

6. Created GaussianNB() object to implement Naive Bayes algorithm.

7. Found predictions using X_Test data.

8. Evaluated the model by finding the accuracy score for the test data.

9. Got the classification report for the test and predicated data.

```
X_train, X_test, Y_train, Y_test= train_test_split(x, y, test_size=0.4, random_state=0)

model = GaussianNB()
model.fit(X_train,Y_train)

y_pred = model.predict(X_test)


score = accuracy_score(Y_test,y_pred)*100
print("accuracy score: " + str(score))

print(classification_report(Y_test, y_pred))
```

```
accuracy score: 0.4454342984409799
              precision    recall  f1-score   support

       55993       0.00      0.00      0.00         1
       60000       0.00      0.00      0.00         1
       67000       0.00      0.00      0.00         1
       73000       0.00      0.00      0.00         1
       75000       0.00      0.00      0.00         1
       76000       0.00      0.00      0.00         1
       78000       0.00      0.00      0.00         1
       79000       0.00      0.00      0.00         1
       79500       0.00      0.00      0.00         1
       80000       0.00      0.00      0.00         2
       81000       0.00      0.00      0.00         1
       82500       0.00      0.00      0.00         1
       83000       0.00      0.00      0.00         1
       85000       0.00      0.00      0.00         1
       85400       0.00      0.00      0.00         1
       86000       0.00      0.00      0.00         2
```

10. Now calculated the score using KNN algorithm

```
#KNN
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors = 5)
knn.fit(X_train,Y_train)
score = knn.score(X_test,Y_test)
print("KNN socre: " + str(score))
```

```
KNN socre: 0.004454342984409799
```

11. Calculated the score also using SVM linear model for inear kernel rdf kernel.

<u>Linear:</u>

```
#SVC Linear
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.svm import SVC

svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, Y_train)

y_pred = svclassifier.predict(X_test)
# print(confusion_matrix(Y_test,y_pred))
print(classification_report(Y_test,y_pred))
```

```
              precision    recall  f1-score   support

       55993       0.00      0.00      0.00         1
       60000       0.00      0.00      0.00         1
       67000       0.00      0.00      0.00         1
       73000       0.00      0.00      0.00         1
       75000       0.00      0.00      0.00         1
       76000       0.00      0.00      0.00         1
       78000       0.00      0.00      0.00         1
       79000       0.00      0.00      0.00         1
       79500       0.00      0.00      0.00         1
       79900       0.00      0.00      0.00         0
       80000       0.00      0.00      0.00         2
       81000       0.00      0.00      0.00         1
       82000       0.00      0.00      0.00         0
       82500       0.00      0.00      0.00         1
       83000       0.00      0.00      0.00         1
       85000       0.00      0.00      0.00         1
       85400       0.00      0.00      0.00         1
```

Non-Linear:

```
#SVC linear
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.svm import SVC

svclassifier = SVC()
svclassifier.fit(X_train, Y_train)

y_pred1 = svclassifier.predict(X_test)

print(classification_report(Y_test,y_pred))
```

```
              precision    recall  f1-score   support

       55993       0.00      0.00      0.00         1
       60000       0.00      0.00      0.00         1
       67000       0.00      0.00      0.00         1
       73000       0.00      0.00      0.00         1
       75000       0.00      0.00      0.00         1
       76000       0.00      0.00      0.00         1
       78000       0.00      0.00      0.00         1
       79000       0.00      0.00      0.00         1
       79500       0.00      0.00      0.00         1
       79900       0.00      0.00      0.00         0
       80000       0.00      0.00      0.00         2
       81000       0.00      0.00      0.00         1
       82000       0.00      0.00      0.00         0
       82500       0.00      0.00      0.00         1
       83000       0.00      0.00      0.00         1
       85000       0.00      0.00      0.00         1
       85400       0.00      0.00      0.00         1
       86000       0.00      0.00      0.00         2
```

12. We found there is very slight difference after evaluating SVM for linear and rdf kernel.

DATASETS:

Link for datasets: https://drive.google.com/drive/u/0/folders/1IRP9UlclTDTCKO1XD6gKb78ZNJBFCLAR

PARAMETERS:

Target elements in individual questions are our considered parameters.

- Question 1: class
- Question 2: In this we have done clustering
- Question 3: temperature
- Question 4: We considered class and text features
- Question 5: saleprice

EVALUATION AND DISCUSSION:

<u>Question 1:</u>

i)we have used both naive baes and KNN classification algorithms.

ii)The accuracy score using naive baes algorithm is 99.320 and the score with KNN is 99.83

iii)We have also inferred that there are 2 classes 0 and 1 and displayed samples in both the classes.

<u>Question 2:</u>

i)Using elbow method, we inferred that the optimized K value is 5.

ii)The silhouette score is 0.55, from this we can say that the model is good.

iii)Through the visualization also we can see that there are 5 clusters

<u>Question 3:</u>

i)We have plotted different columns like temperature and humidity and we found that there are variations.

ii)we found the top 5 correlated values.

iii)From R2 square=0.99 which almost equals to 1, we inferred that model is very close to the fit line.

<u>Question 4:</u>

i)Using count Vectorizer, we analyzed the shape of the wordcount.

ii)Found the accuracy score for three classifications using pipeline.

<u>Question 5:</u>

i)We have applied three classification algorithms and inferred that Naive bayes classifier better result.

ii)We also found that SVM with linear gives better performance.

CONCLUSION:

This project helped us learning and implementing different algorithms**.** We have analysed all the algorithms by calculating accuracy score and successfully implemented all algorithms.

GITHUB REFERENCE: https://github.com/PallaviArikatla/Python_Exams/wiki/Project_Exam_1

VIDEO REFERENCE:
https://umkc.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=7a6e6506-6d88-478b-9d48-abf0017043f9