My grades for **Assignment 2**

Q1

8.5 / 10

Submit here the pdf printout of your entire notebook, including the outputs of your code.

Gajjala_Jahnavi_Qiu_Sichen_Kasibhatla_Venkata_Sreenivas_Prasad_Assi

February 13, 2024

1 CSCI 4146/6409 - Process of Data Science

1.1 Assignment 2

Winter 2024

Due on: 11 February 2024, 23:59 ADT

Please delete the instructions from your submitted notebook, but keep the original questions.

1.2 Read Before You Star

We will be using Crowdmark this term for submitting and marking assignments. Crowdmark allows markers to provide specific comments on submitted documents, leading to better quality feedback.

1. You will **receive an email** from the Crowdmark mailer with a button "Go to assignment" and a link to use if the button does not work for you.
2. You need to submit your answer to **each question separately**.
3. Your answer to a math question should be as **Image/PDF file**. It can be a photo of a handwritten solution or a pdf file of a typeset solution. The choice is up to you. The format will not affect your mark, as long as handwriting is neat and very readable. Please note that your submitted image **should not be rotated**. Rotaded images will not be marked, but they will automatically receive a zero.
4. Your answer to a coding/experimentation question will be **in two parts** (i.e. two questions). The first part is an **Image/PDF answer**, where you upload the PDF printout of your python notebook (pdf file). The second part is a text answer question, to which you attach your python notebook (ipynb file). Please do not zip any of the files you upload to Crowdmark.

1.3 Submitting your solution as a group

You are allowed to work on the assignment individually or as a group of 3 students. Here are the crowdmark rules for a group submition:

Once the assignment has been distributed, students will receive an email with a link to view the assignment.

- Students will be able to add or edit group members before submitting.
- Students can only add members who have not already been added to another group. Once a student is added to a group, the submission page is shared among the group members and any member can submit work on behalf of the group.

- Students will receive an email notification that they have been added to a group by a particular user.
- Students also have access to the submission log in their portflio. The log includes the addition or removal of group members and submission activity for each individual user.
- Once a group assignment has been submitted, students will not be able to edit the group members. Students will need to contact their instructor to make any changes to group members after submission. Note: if students have submitted individually, moving students to a group will clear their individual submissions.

1.4 Instructions for generating a PDF printout of a Jupyter notebook.

There are 3 ways to save a Jupyter Notebook as a PDF file: 1. Using Jupyter Notebook 2. Using `nbconvert` Command-Line Tool. 3. Using print to a PDF file in the browser (**DON'T USE**)

1. **Using Jupyter Notebook**
   1. Install Jupyter Notebook. If you install Anaconda, Jupyter Notebook will be included in the Anaconda Navigator.
   2. Open the notebook in Jupyter.
   3. Select "File" from the menu bar, then "Download as", then "PDF via LaTeX". See the image below.
   4. The notebook will be converted to a PDF file and downloaded to your computer.

2. **Using `nbconvert` Command-Line Tool**
   1. Install nbconvert.
   2. Open the terminal and navigate to the folder containing the notebook.
   3. Enter the following command: `jupyter nbconvert --to pdf <notebook filename>`.
   4. The notebook will be converted to a PDF file and saved in the same folder.
3. **Using print to a PDF file in the browser (DON'T USE)**
   - The produced PDFs can't be parsed and hard to mark, so please don't use this method. If for any reasons you have a hard time with producing a PDF don't hasitate to contact the TA.

Note: If you don't want to install any software localy, you can use a Dal FCS-hosted Jupyter Web Interface Service: https://timberlea.cs.dal.ca:8000/. Should you run into any errors, please ask the CS HelpDesk for help.

Note: in solving the math questions, aim for general (symbolic) solutions and substitute the specific numbers at the end. This demonstrates a solid understanding of the key concepts. You can answer the math questions in two ways: * **Use LaTeX to typeset the equations**. Section H of this LaTeX reference sheet is a good reference. Here is another LaTeX reference sheet. The equations in the questions are typeset in LaTeX, so you can use them as examples. * **Use neat handwriting**, scan your solution using AdobeScan, or Dropbox on your mobile phone, and upload the image file to the corresponding Crowdmark Image/PDF question.

1.5 Submission Instructions

Your answers to the questions should be in a solution notebook, in the form of code and text cells, using markdown for your text responses. **You should also include in the notebook the results of running your code. This means that you must not clear the output produced by your program**.

Need help with markdown? Check Here!

The marking criteria are described in rubrics in the syllabus.

You can submit multiple editions of your assignment. Only the last one will be marked. It is recommended to upload a complete submission, even if you are still improving it, so that you have something into the system if your computer fails for whatever reason.

> **IMPORTANT**: PLEASE NAME YOUR PYTHON NOTEBOOK FILE AS: * `<LAST_NAME>-<FIRST_NAME>_<PARTNER_LAST_NAME>-<PARTNER_FIRST_NAME>-Assignment-N.ipynb`

for example: **Milios-Evangelos-Assignment-2.ipynb** \

2 Predictive Modelling

In this assignment, we will continue using the real estate dataset from Assignment 1 (House Prices Prediction Kaggle competition). Now that you have a better understanding of the business objective and its connection to the data, we will proceed to the next step in the process of data science: Predictive Modelling. For this assignment, we will create a new column called '**SalePriceCategory**,' to serve as our **target variable** for prediction.

Link to the dataset: Kaggle Dataset.

[ ]:
```
# Importing necessary packages
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
pd.options.plotting.backend = "plotly"

data = pd.read_csv('train.csv')
data['SalePriceCategory'] = pd.qcut(data.SalePrice, q=5, labels=['Very Low',␣
↪'Low', 'Medium', 'High', 'Very High']) # Bins the SalePrice into 5␣
↪equal-sized buckets.
```

[ ]:
```
# Distribution of SalePriceCategory
data['SalePriceCategory'].hist(title='Distribution of SalePriceCategory')
```

2.1 Q1. [0.1] Task Formulation

1. Explain the task you are solving (e.g., supervised/unsupervised, classification/regression/clustering, or similarity matching, etc.)?

This is a supervised classification task. In supervised learning, the Machine learning model is trained on a labeled dataset, which means it learns from data that already contains the target variable with labels ('SalePriceCategory' in this case). The goal is to classify the category of the

predicted sale price from a set of features describing the house. The 'SalePriceCategory' acts as the target variable, which has been divided into discrete categories (Very Low, Low, Medium, High, Very High). The objective of the model is to use the patterns it has learnt from the training data to classify unseen houses into these predetermined groups.

2.2 Q2. [0.2] Feature Correlation

1. Split the dataset into a training set and a test set using a 70:30 ratio. You can achieve this by utilizing the train_test_split utility function from scikit-learn. Ensure a similar distribution of categories in both the training and test sets by using the stratify parameter in train_test_split.

[ ]:
```
# Generating a data quality report for continuous features in the DataFrame.
continuous_report = data.describe(include=['float64', 'int64'])
continuous_report
```

[ ]:
```
                Id   MSSubClass  LotFrontage        LotArea  OverallQual  \
count  1460.000000  1460.000000  1201.000000    1460.000000  1460.000000
mean    730.500000    56.897260    70.049958   10516.828082     6.099315
std     421.610009    42.300571    24.284752    9981.264932     1.382997
min       1.000000    20.000000    21.000000    1300.000000     1.000000
25%     365.750000    20.000000    59.000000    7553.500000     5.000000
50%     730.500000    50.000000    69.000000    9478.500000     6.000000
75%    1095.250000    70.000000    80.000000   11601.500000     7.000000
max    1460.000000   190.000000   313.000000  215245.000000    10.000000

       OverallCond    YearBuilt  YearRemodAdd   MasVnrArea   BsmtFinSF1  …  \
count  1460.000000  1460.000000   1460.000000  1452.000000  1460.000000  …
mean      5.575342  1971.267808   1984.865753   103.685262   443.639726  …
std       1.112799    30.202904     20.645407   181.066207   456.098091  …
min       1.000000  1872.000000   1950.000000     0.000000     0.000000  …
25%       5.000000  1954.000000   1967.000000     0.000000     0.000000  …
50%       5.000000  1973.000000   1994.000000     0.000000   383.500000  …
75%       6.000000  2000.000000   2004.000000   166.000000   712.250000  …
max       9.000000  2010.000000   2010.000000  1600.000000  5644.000000  …

        WoodDeckSF  OpenPorchSF  EnclosedPorch    3SsnPorch  ScreenPorch  \
count  1460.000000  1460.000000    1460.000000  1460.000000  1460.000000
mean     94.244521    46.660274      21.954110     3.409589    15.060959
std     125.338794    66.256028      61.119149    29.317331    55.757415
min       0.000000     0.000000       0.000000     0.000000     0.000000
25%       0.000000     0.000000       0.000000     0.000000     0.000000
50%       0.000000    25.000000       0.000000     0.000000     0.000000
75%     168.000000    68.000000       0.000000     0.000000     0.000000
max     857.000000   547.000000     552.000000   508.000000   480.000000

          PoolArea       MiscVal       MoSold       YrSold      SalePrice
count  1460.000000   1460.000000  1460.000000  1460.000000    1460.000000
```

```
mean      2.758904     43.489041     6.321918  2007.815753  180921.195890
std      40.177307    496.123024     2.703626     1.328095   79442.502883
min       0.000000      0.000000     1.000000  2006.000000   34900.000000
25%       0.000000      0.000000     5.000000  2007.000000  129975.000000
50%       0.000000      0.000000     6.000000  2008.000000  163000.000000
75%       0.000000      0.000000     8.000000  2009.000000  214000.000000
max     738.000000  15500.000000    12.000000  2010.000000  755000.000000

[8 rows x 38 columns]
```

[ ]:
```
# Generating a data quality report for categorical features in the DataFrame.
category_report = data.describe(include=['category', 'object'])
category_report
```

[ ]:
```
       MSZoning Street Alley LotShape LandContour Utilities LotConfig  \
count      1460   1460    91     1460        1460      1460      1460
unique        5      2     2        4           4         2         5
top          RL   Pave  Grvl      Reg         Lvl    AllPub    Inside
freq       1151   1454    50      925        1311      1459      1052

       LandSlope Neighborhood Condition1  … GarageFinish GarageQual  \
count       1460         1460       1460  …         1379       1379
unique         3           25          9  …            3          5
top          Gtl        NAmes       Norm  …          Unf         TA
freq        1382          225       1260  …          605       1311

       GarageCond PavedDrive PoolQC  Fence MiscFeature SaleType SaleCondition  \
count        1379       1460      7    281          54     1460          1460
unique          5          3      3      4           4        9             6
top            TA          Y     Gd  MnPrv        Shed       WD        Normal
freq         1326       1340      3    157          49     1267          1198

       SalePriceCategory
count               1460
unique                 5
top             Very Low
freq                 295

[4 rows x 44 columns]
```

[ ]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 82 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Id                 1460 non-null   int64
```

please do not dump long output in the report

```
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
 25  MasVnrType         1452 non-null   object
 26  MasVnrArea         1452 non-null   float64
 27  ExterQual          1460 non-null   object
 28  ExterCond          1460 non-null   object
 29  Foundation         1460 non-null   object
 30  BsmtQual           1423 non-null   object
 31  BsmtCond           1423 non-null   object
 32  BsmtExposure       1422 non-null   object
 33  BsmtFinType1       1423 non-null   object
 34  BsmtFinSF1         1460 non-null   int64
 35  BsmtFinType2       1422 non-null   object
 36  BsmtFinSF2         1460 non-null   int64
 37  BsmtUnfSF          1460 non-null   int64
 38  TotalBsmtSF        1460 non-null   int64
 39  Heating            1460 non-null   object
 40  HeatingQC          1460 non-null   object
 41  CentralAir         1460 non-null   object
 42  Electrical         1459 non-null   object
 43  1stFlrSF           1460 non-null   int64
 44  2ndFlrSF           1460 non-null   int64
 45  LowQualFinSF       1460 non-null   int64
 46  GrLivArea          1460 non-null   int64
 47  BsmtFullBath       1460 non-null   int64
 48  BsmtHalfBath       1460 non-null   int64
```

```
 49  FullBath           1460 non-null   int64
 50  HalfBath           1460 non-null   int64
 51  BedroomAbvGr       1460 non-null   int64
 52  KitchenAbvGr       1460 non-null   int64
 53  KitchenQual        1460 non-null   object
 54  TotRmsAbvGrd       1460 non-null   int64
 55  Functional         1460 non-null   object
 56  Fireplaces         1460 non-null   int64
 57  FireplaceQu        770 non-null    object
 58  GarageType         1379 non-null   object
 59  GarageYrBlt        1379 non-null   float64
 60  GarageFinish       1379 non-null   object
 61  GarageCars         1460 non-null   int64
 62  GarageArea         1460 non-null   int64
 63  GarageQual         1379 non-null   object
 64  GarageCond         1379 non-null   object
 65  PavedDrive         1460 non-null   object
 66  WoodDeckSF         1460 non-null   int64
 67  OpenPorchSF        1460 non-null   int64
 68  EnclosedPorch      1460 non-null   int64
 69  3SsnPorch          1460 non-null   int64
 70  ScreenPorch        1460 non-null   int64
 71  PoolArea           1460 non-null   int64
 72  PoolQC             7 non-null      object
 73  Fence              281 non-null    object
 74  MiscFeature        54 non-null     object
 75  MiscVal            1460 non-null   int64
 76  MoSold             1460 non-null   int64
 77  YrSold             1460 non-null   int64
 78  SaleType           1460 non-null   object
 79  SaleCondition      1460 non-null   object
 80  SalePrice          1460 non-null   int64
 81  SalePriceCategory  1460 non-null   category
dtypes: category(1), float64(3), int64(35), object(43)
memory usage: 925.7+ KB
```

[ ]:
```
# Data cleaning - handling missing values

# Dropping the columns with majority Nan values
'''
Alley              91 non-null     object
PoolQC             7 non-null      object
Fence              281 non-null    object
MiscFeature        54 non-null     object

It indicates that there is an inadequate feature representation in the dataset
for these features.
```

```
Features like these might not provide the model with sufficient information to
learn from.

'''
data.drop(['Alley','PoolQC', 'Fence', 'MiscFeature'],axis=1, inplace = True)

# Filling Missing Values in Categorical Columns with Mode
'''
The most frequent value (mode) in each corresponding column is used to fill
in any missing values in categorical columns. This method is useful for
categorical data because it avoids introducing significant bias and maintains
the current distribution of categories

'''
categorical_cols = data.select_dtypes(include=['object', 'category']).columns
for i in categorical_cols:
  data[i].fillna(data[i].mode()[0], inplace=True)

# Filling Missing Values in Numerical Columns with Mean
'''
The mean value of each corresponding column is used to fill in any missing
values in numerical columns. A measure of central tendency that can
preserve the feature's overall distribution is provided by the mean.

'''
float_cols = data.select_dtypes(include=['float64', 'int64']).columns
for i in float_cols:
  data[i].fillna(data[i].mean(), inplace=True)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 78 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        1460 non-null   float64
 4   LotArea            1460 non-null   int64
 5   Street             1460 non-null   object
 6   LotShape           1460 non-null   object
 7   LandContour        1460 non-null   object
 8   Utilities          1460 non-null   object
 9   LotConfig          1460 non-null   object
 10  LandSlope          1460 non-null   object
 11  Neighborhood       1460 non-null   object
 12  Condition1         1460 non-null   object
```

please do not dump long output in the report

```
13  Condition2         1460 non-null   object
14  BldgType           1460 non-null   object
15  HouseStyle         1460 non-null   object
16  OverallQual        1460 non-null   int64
17  OverallCond        1460 non-null   int64
18  YearBuilt          1460 non-null   int64
19  YearRemodAdd       1460 non-null   int64
20  RoofStyle          1460 non-null   object
21  RoofMatl           1460 non-null   object
22  Exterior1st        1460 non-null   object
23  Exterior2nd        1460 non-null   object
24  MasVnrType         1460 non-null   object
25  MasVnrArea         1460 non-null   float64
26  ExterQual          1460 non-null   object
27  ExterCond          1460 non-null   object
28  Foundation         1460 non-null   object
29  BsmtQual           1460 non-null   object
30  BsmtCond           1460 non-null   object
31  BsmtExposure       1460 non-null   object
32  BsmtFinType1       1460 non-null   object
33  BsmtFinSF1         1460 non-null   int64
34  BsmtFinType2       1460 non-null   object
35  BsmtFinSF2         1460 non-null   int64
36  BsmtUnfSF          1460 non-null   int64
37  TotalBsmtSF        1460 non-null   int64
38  Heating            1460 non-null   object
39  HeatingQC          1460 non-null   object
40  CentralAir         1460 non-null   object
41  Electrical         1460 non-null   object
42  1stFlrSF           1460 non-null   int64
43  2ndFlrSF           1460 non-null   int64
44  LowQualFinSF       1460 non-null   int64
45  GrLivArea          1460 non-null   int64
46  BsmtFullBath       1460 non-null   int64
47  BsmtHalfBath       1460 non-null   int64
48  FullBath           1460 non-null   int64
49  HalfBath           1460 non-null   int64
50  BedroomAbvGr       1460 non-null   int64
51  KitchenAbvGr       1460 non-null   int64
52  KitchenQual        1460 non-null   object
53  TotRmsAbvGrd       1460 non-null   int64
54  Functional         1460 non-null   object
55  Fireplaces         1460 non-null   int64
56  FireplaceQu        1460 non-null   object
57  GarageType         1460 non-null   object
58  GarageYrBlt        1460 non-null   float64
59  GarageFinish       1460 non-null   object
60  GarageCars         1460 non-null   int64
```

```
 61  GarageArea         1460 non-null   int64
 62  GarageQual         1460 non-null   object
 63  GarageCond         1460 non-null   object
 64  PavedDrive         1460 non-null   object
 65  WoodDeckSF         1460 non-null   int64
 66  OpenPorchSF        1460 non-null   int64
 67  EnclosedPorch      1460 non-null   int64
 68  3SsnPorch          1460 non-null   int64
 69  ScreenPorch        1460 non-null   int64
 70  PoolArea           1460 non-null   int64
 71  MiscVal            1460 non-null   int64
 72  MoSold             1460 non-null   int64
 73  YrSold             1460 non-null   int64
 74  SaleType           1460 non-null   object
 75  SaleCondition      1460 non-null   object
 76  SalePrice          1460 non-null   int64
 77  SalePriceCategory  1460 non-null   category
dtypes: category(1), float64(3), int64(35), object(39)
memory usage: 880.0+ KB
```

[ ]:
```
y=pd.DataFrame(data['SalePriceCategory'])
# Identifying categorical columns (excluding the target variable)

dropped = data.drop(['SalePrice','SalePriceCategory'],axis=1)
categorical_cols = dropped.select_dtypes(include=['object', 'category']).columns

# Apply one-hot encoding to convert categorical features into numeric
'''
When there is no inherent ordering in the categories, one-hot encoding works
well for nominal categorical data. Without enforcing any ordinality, one-hot
encoding preserves the categorical aspect of the data by generating distinct
binary (0 or 1) columns for each category.

One-hot encoding of categorical data improves the performance of tree-based
and linear models. This is due to the fact that when ordinal encodings, such
as label encoding, are applied to nominal data, these models, which are
sensitive to numerical values, may misinterpret the numerical relationships.

'''
data_encoded = pd.get_dummies(dropped, columns=categorical_cols)
data_encoded
```

[ ]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	\
0	1	60	65.0	8450	7	5	
1	2	20	80.0	9600	6	8	
2	3	60	68.0	11250	7	5	
3	4	70	60.0	9550	7	5	

```
4        5          60         84.0    14260            8            5
…     …          …          …      …          …           …
1455  1456          60         62.0     7917            6            5
1456  1457          20         85.0    13175            6            6
1457  1458          70         66.0     9042            7            9
1458  1459          20         68.0     9717            5            6
1459  1460          20         75.0     9937            5            6

      YearBuilt  YearRemodAdd  MasVnrArea  BsmtFinSF1  …  SaleType_ConLw  \
0          2003          2003       196.0         706  …               0
1          1976          1976         0.0         978  …               0
2          2001          2002       162.0         486  …               0
3          1915          1970         0.0         216  …               0
4          2000          2000       350.0         655  …               0
…         …           …          …          …  …              …
1455       1999          2000         0.0           0  …               0
1456       1978          1988       119.0         790  …               0
1457       1941          2006         0.0         275  …               0
```

```
1458       1950          1996         0.0          49  …                0
1459       1965          1965         0.0         830  …                0

      SaleType_New  SaleType_Oth  SaleType_WD  SaleCondition_Abnorml  \
0                0             0            1                      0
1                0             0            1                      0
2                0             0            1                      0
3                0             0            1                      1
4                0             0            1                      0
…              …             …          …                      …
1455             0             0            1                      0
1456             0             0            1                      0
1457             0             0            1                      0
1458             0             0            1                      0
1459             0             0            1                      0

      SaleCondition_AdjLand  SaleCondition_Alloca  SaleCondition_Family  \
0                         0                     0                     0
1                         0                     0                     0
2                         0                     0                     0
3                         0                     0                     0
4                         0                     0                     0
…                       …                     …                     …
1455                      0                     0                     0
1456                      0                     0                     0
1457                      0                     0                     0
1458                      0                     0                     0
1459                      0                     0                     0
```



```
      SaleCondition_Normal  SaleCondition_Partial
0                        1                      0
1                        1                      0
2                        1                      0
3                        0                      0
4                        1                      0
…                      …                      …
1455                     1                      0
1456                     1                      0
1457                     1                      0
1458                     1                      0
1459                     1                      0

[1460 rows x 276 columns]
```

[ ]:
```
#Splitting the dataset into train and test data
X=data_encoded
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3,␣
↪random_state=42, stratify=y)
```

2. Select the **top 10** features with the highest correlation to the **SalePrice** feature. Ensure that only the training set is used in this step.

[ ]:
```
# Calculate correlations
correlations = X_train.corrwith(data.SalePrice).abs().
↪sort_values(ascending=False)
print(correlations)
# Select top 10 features
top_10_features = correlations.head(10).index.tolist()
X_train_top_10 = X_train[top_10_features]
top_10_features
```

```
OverallQual          0.796416
GrLivArea            0.715978
GarageArea           0.647604
GarageCars           0.642288
```

```
TotalBsmtSF           0.613887
                         …
Exterior2nd_Other         NaN
Heating_OthW              NaN
Electrical_Mix            NaN
GarageQual_Ex             NaN
GarageCond_Ex             NaN
Length: 276, dtype: float64
```

[ ]:
```
['OverallQual',
 'GrLivArea',
 'GarageArea',
```

```
 'GarageCars',
 'TotalBsmtSF',
 'ExterQual_TA',
 '1stFlrSF',
 'FullBath',
 'GarageFinish_Unf',
 'TotRmsAbvGrd']
```

2.3 Q3. [0.3] Hyperparameter Tuning

1. Utilize only the top 10 features selected from Q2. Train a Decision Tree Classifier Model with the default hyperparameters to predict the 'SalePriceCategory' and report the accuracy score.

[ ]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Train the classifier
clf = DecisionTreeClassifier()
clf.fit(X_train_top_10, y_train)

X_test = X_test[top_10_features]

# Calculate accuracy
y_test_pred = clf.predict(X_test)
train_accuracy = accuracy_score(y_test, y_test_pred)
train_accuracy
```

[ ]:
```
0.593607305936073
```

2. Read the documentation and utilize the RandomizedSearchCV method from scikit-learn to perform hyperparameter tuning.

   Note: We do not use the test set at this stage.

(Various methods exist for finding the best hyperparameters for your model. You can read about them here.)

[ ]:
```
from sklearn.model_selection import RandomizedSearchCV
import numpy as np

# Define parameter grid
param_dist = {
    'criterion': ['gini','entropy'],
    'max_depth': [None] + list(np.arange(1, 20)),
    'min_samples_split': np.arange(2, 20),
    'min_samples_leaf': np.arange(1, 20),
}
```

```
# Initialize RandomizedSearchCV
random_search = RandomizedSearchCV(clf, param_distributions=param_dist,␣
↪n_iter=100, random_state=42, cv=5, verbose=1, n_jobs=-1)

# Fit RandomizedSearchCV
random_search.fit(X_train_top_10, y_train)

# Best parameters
best_params = random_search.best_params_
```

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits
```

[ ]:
```
best_params
```

[ ]:
```
{'min_samples_split': 5,
 'min_samples_leaf': 3,
 'max_depth': 7,
 'criterion': 'gini'}
```

2.4 Q4. [0.4] Model Evaluation

1. Utilize the model with the best hyperparameters found in Q3 and evaluate it on the test set.

[ ]:
```
# Use the best parameters to fit the classifier
best_clf = DecisionTreeClassifier(**best_params, random_state=42)
best_clf.fit(X_train_top_10, y_train)

# Predict on test set
```

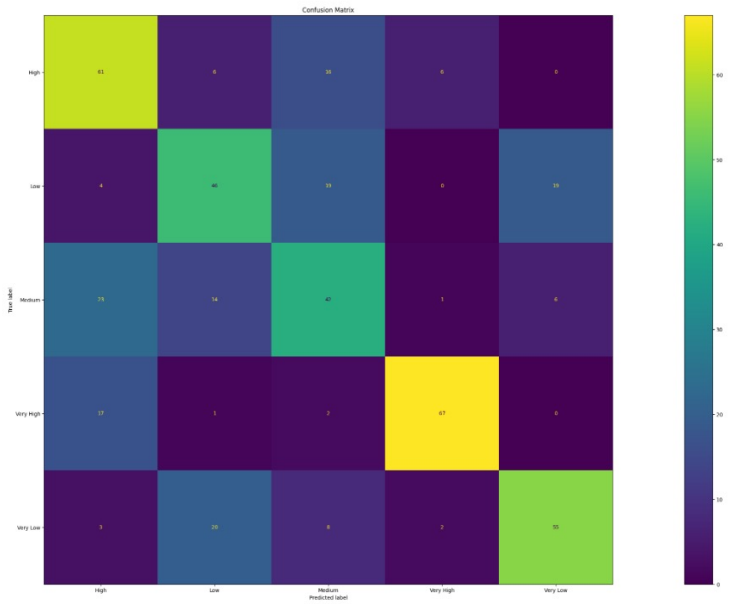
```
X_test_top_10 = X_test[top_10_features]
y_test_pred = best_clf.predict(X_test_top_10)

# Calculate accuracy
test_accuracy = accuracy_score(y_test, y_test_pred)
test_accuracy
```

[ ]: 0.6187214611872146

2. Plot the confusion matrix of the best model using the ConfusionMatrixDisplay utility function from scikit-learn. Analyze the model's performance based on the evaluation metric.

result analysis is missing

[ ]:
```
from sklearn.metrics import ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Plot the Confusion matrix
ConfusionMatrixDisplay.from_predictions(y_test, y_test_pred)
plt.title('Confusion Matrix')
plt.show()
```

[ ]:
```
# Evaluate the performance of a classifier
from sklearn.metrics import classification_report

print(classification_report(y_test, y_test_pred))
```

```
              precision    recall  f1-score   support

        High       0.56      0.69      0.62        89
         Low       0.53      0.52      0.53        88
      Medium       0.48      0.49      0.49        86
   Very High       0.88      0.77      0.82        87
    Very Low       0.69      0.62      0.65        88

    accuracy                           0.62       438
   macro avg       0.63      0.62      0.62       438
weighted avg       0.63      0.62      0.62       438
```

3. Train 10 models using the best hyperparameters from Q3, each with increasing data sizes.

Create a chart with the training data size on the x-axis, indicating the size used for training the model. Represent both the train and test accuracy scores on the y-axis. Analyze the results.

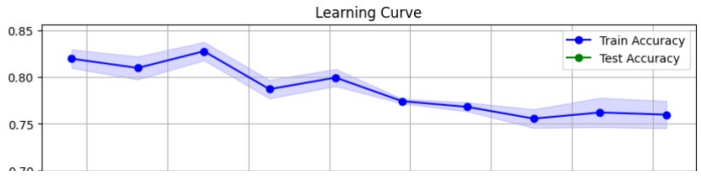
(You may prefer to use the LearningCurveDisplay utility function from scikit-learn, which uses 5-fold cross-validation by default. You can use *score_type="both"* to plot both train and test performance.)

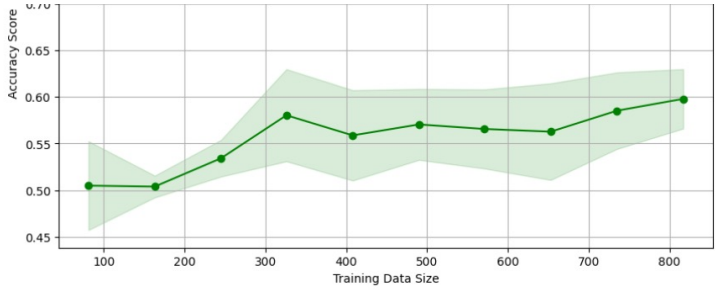
[ ]:
```
from sklearn.model_selection import learning_curve
import matplotlib.pyplot as plt
import numpy as np

# Best hyperparameters from Q3
best_hyperparams = best_params

# Initialize the model with the best hyperparameters
model = DecisionTreeClassifier(**best_hyperparams, random_state=42)

# Define training sizes
train_sizes = np.linspace(0.1, 1.0, 10)

# Compute the learning curves
train_sizes, train_scores, test_scores = learning_curve(
    model, X_train_top_10, y_train, cv=5, train_sizes=train_sizes,␣
 ↪scoring='accuracy', n_jobs=-1)

# Calculate mean and standard deviation for train and test scores
train_mean = np.mean(train_scores, axis=1)
```

```
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

# Plot learning curves
plt.figure(figsize=(10, 6))
plt.plot(train_sizes, train_mean, label='Train Accuracy', color='blue',␣
 ↪marker='o')
plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std,␣
 ↪color='blue', alpha=0.15)

plt.plot(train_sizes, test_mean, label='Test Accuracy', color='green',␣
 ↪marker='o')
plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std,␣
 ↪color='green', alpha=0.15)

plt.title('Learning Curve')
plt.xlabel('Training Data Size')
plt.ylabel('Accuracy Score')
```

16

```
plt.legend()
plt.grid()
plt.show()
```





The presented learning curve illustrates the relationship between the size of the training dataset and the machine learning model's accuracy on the training and test sets.

Training Accuracy: The training accuracy starts high and remains relatively stable as more data is added. This implies that the model fits the training data well, which may mean that the model's complexity is suitable for the underlying patterns in the data.

Testing Accuracy: When the training data size is minimal, the test accuracy begins to decline, suggesting that the model does not generalize well with tiny amounts of data. However, the test accuracy gradually rises with increasing training data, indicating that the model's capacity for generalization is getting better with increased data.

Convergence: As the size of the training data increases, the two curves are converging. It indicates that there is less overfitting since the performance difference between training and testing is decreasing.

[ ]:

Clarity b-accepta

S&D b-MInimal 2

Func a-Excellent 5

Please provide effective and meaningful comments when appropriate.

Please do not dump long output in the report

Q2

0 / 0

Submit here your Python notebook (ipynb) as an attachment

Gajjala-Jahnavi_Qiu-Sichen_Kasibhatla-Venkata Sreenivas Prasad-Assignment-2.ipynb