In [17]:
```
import numpy as np
import pandas as pd
import matplotlib.pylab as plt
%matplotlib inline
```

Assignment 5

1. Choose a regression dataset (bikeshare is allowed), perform a test/train split, and build a regression model (just like in assingnment 3), and calculate the

Training Error (MSE, MAE)

Testing Error (MSE, MAE)

In [2]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
```

In [3]:
```
# Training dataset
df = pd.read_csv('../data/WineQT.csv')
```

Dataset: https://www.kaggle.com/rajyellow46/wine-quality

In [4]:
```
df.columns
```

Out[4]:
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

In [5]:
```
y = df["quality"]
x = df.drop(["quality"], axis = 1)
```

In [6]:
```
y.shape, y.size
```

Out[6]:
```
((1142,), 1142)
```

In [7]:
```
x.shape, x.size
```

Out[7]:
```
((1142, 11), 12562)
```

In [8]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.5)
```

In [9]:
```
linreg = LinearRegression()
linreg.fit(x_train, y_train)
linreg.coef_, linreg.intercept_
```

Out[9]:
```
(array([ 3.73731261e-02, -8.27295822e-01, -1.14744117e-01,  1.70232093e-02,
        -2.02425202e+00,  7.27119711e-04, -2.80587246e-03, -2.37449383e+01,
        -1.91956226e-01,  9.48032814e-01,  2.73168379e-01]),
 26.91899784410421)
```

In [10]:
```
pred = linreg.predict(x_train)
```

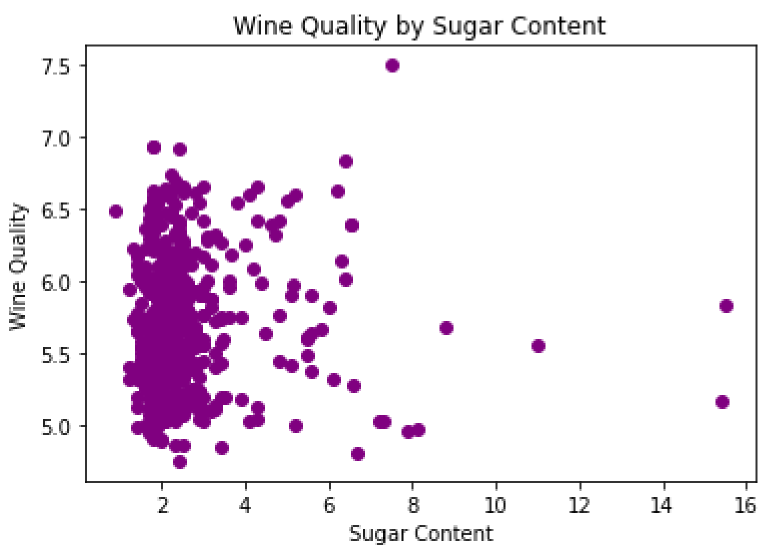
In [11]:
```
sugar = x_train["residual sugar"]
```

In [12]:
```
sugar.shape, pred.shape
```

Out[12]:
```
((571,), (571,))
```

In [13]:
```
plt.scatter(sugar, pred, c = "purple")
plt.title("Wine Quality by Sugar Content")
plt.xlabel("Sugar Content")
plt.ylabel("Wine Quality")
```

Out[13]:
```
Text(0, 0.5, 'Wine Quality')
```



In [14]:
```
# Train MSE and MAE
print(mean_squared_error(y_train, pred))
print(mean_absolute_error(y_train, pred))
```

```
0.4260571458135494
0.5103115138055352
```

In [15]:
```
# Test MSE and MAE
print(mean_squared_error(y_test, np.dot(x_test, linreg.coef_) + linreg.intercept_))
print(mean_absolute_error(y_test, np.dot(x_test, linreg.coef_) + linreg.intercept_))
```

```
0.3947245336824352
0.4845559304017829
```

2. Choose a classification dataset (not the adult.data set, perform test/train split and create a classification model (your choice but DecisionTree is fine). Calculate:

Accuracy

Confusion Matrix

Classifcation Report

In [186…

```python
from sklearn import preprocessing
```

In [187…

```python
new_df = pd.read_csv('../data/Loan_Default.csv')
new_df.dropna()
```

Out[187]:

	ID	year	loan_limit	Gender	approv_in_adv	loan_type	loan_purpose	Credit_Worthine
2	24892	2019	cf	Male	pre	type1	p1	
4	24894	2019	cf	Joint	pre	type1	p1	
5	24895	2019	cf	Joint	pre	type1	p1	
6	24896	2019	cf	Joint	pre	type1	p3	
8	24898	2019	cf	Joint	nopre	type1	p3	
...	...	...	...	...	...	...	...	
148665	173555	2019	cf	Sex Not Available	nopre	type1	p3	
148666	173556	2019	cf	Male	nopre	type1	p1	
148667	173557	2019	cf	Male	nopre	type1	p4	
148668	173558	2019	cf	Female	nopre	type1	p4	
148669	173559	2019	cf	Female	nopre	type1	p3	

98187 rows × 34 columns

In [188…

```
new_df.columns
```

Out[188]:

```
Index(['ID', 'year', 'loan_limit', 'Gender', 'approv_in_adv', 'loan_type',
       'loan_purpose', 'Credit_Worthiness', 'open_credit',
       'business_or_commercial', 'loan_amount', 'rate_of_interest',
       'Interest_rate_spread', 'Upfront_charges', 'term', 'Neg_ammortization',
       'interest_only', 'lump_sum_payment', 'property_value',
       'construction_type', 'occupancy_type', 'Secured_by', 'total_units',
       'income', 'credit_type', 'Credit_Score', 'co-applicant_credit_type',
       'age', 'submission_of_application', 'LTV', 'Region', 'Security_Type',
       'Status', 'dtir1'],
      dtype='object')
```

In [189…

```
le = preprocessing.LabelEncoder()
new_df = new_df.apply(le.fit_transform)
new_df.head()
```

Out[189]:

	ID	year	loan_limit	Gender	approv_in_adv	loan_type	loan_purpose	Credit_Worthiness	open_cr
0	0	0	0	3	0	0	0	0	
1	1	0	0	2	0	1	0	0	
2	2	0	0	2	1	0	0	0	
3	3	0	0	2	0	0	3	0	
4	4	0	0	1	1	0	0	0	

5 rows × 34 columns

In [190…

```
y = new_df["Gender"]
x = new_df.drop(["Gender"], axis = 1)
```

In [191…

```
x.shape, x.size, y.shape, y.size
```

Out[191]:

```
((148670, 33), 4906110, (148670,), 148670)
```

In [208…

```
x.head()
```

Out[208]:

	ID	year	loan_limit	approv_in_adv	loan_type	loan_purpose	Credit_Worthiness	open_credit	bus
0	0	0	0	0	0	0	0	0	
1	1	0	0	0	1	0	0	0	
2	2	0	0	1	0	0	0	0	
3	3	0	0	0	0	3	0	0	
4	4	0	0	1	0	0	0	0	

5 rows × 33 columns

In [209…
```
y.head()
```

Out[209]:
```
0    3
1    2
2    2
3    2
4    1
Name: Gender, dtype: int32
```

Import libraries, define model, test/train/split

In [192…
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import (accuracy_score,
                             classification_report,
                             confusion_matrix
                            )
```

In [193…
```
model = DecisionTreeClassifier(criterion = "entropy")
```

In [194…
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=.50)
```

In [195…
```
x_test.shape, x_train.shape
```

Out[195]:
```
((74335, 33), (74335, 33))
```

In [196…
```
y_test.shape, y_train.shape
```

Out[196]:
```
((74335,), (74335,))
```

Train

In [197…
```
model.fit(x_train, y_train)
```

Out[197]:
```
DecisionTreeClassifier(criterion='entropy')
```

Feature Importances

In [198…
```
list(zip(x.columns, model.feature_importances_))
```

Out[198]:
```
[('ID', 0.06687398709776479),
 ('year', 0.0),
 ('loan_limit', 0.004245246449738155),
 ('approv_in_adv', 0.006383338523583306),
 ('loan_type', 0.007483289087586270 4),
 ('loan_purpose', 0.011938213470781622),
 ('Credit_Worthiness', 0.002402270849811495),
 ('open_credit', 0.0001675651601623388 8),
 ('business_or_commercial', 0.001658236536276381),
 ('loan_amount', 0.040678369821070474),
 ('rate_of_interest', 0.023474122046917782),
 ('Interest_rate_spread', 0.037177713778700355),
 ('Upfront_charges', 0.032489346369188966),
 ('term', 0.009047777113331159),
 ('Neg_ammortization', 0.004458885140066792),
 ('interest_only', 0.0024200429154488446),
 ('lump_sum_payment', 0.002299237074672308),
 ('property_value', 0.0519539761020865),
 ('construction_type', 0.0),
 ('occupancy_type', 0.0038327342741205023),
 ('Secured_by', 0.0),
 ('total_units', 0.0012265809731413667),
 ('income', 0.06103506489583998),
 ('credit_type', 0.011862854438618179),
 ('Credit_Score', 0.06512590211959429),
 ('co-applicant_credit_type', 0.2041935650822844),
 ('age', 0.020413812400228427),
 ('submission_of_application', 0.05629399381045407),
 ('LTV', 0.05280821536757443),
 ('Region', 0.18373752125852552),
 ('Security_Type', 0.0),
 ('Status', 0.0008069582834866927),
 ('dtir1', 0.033511179558944594)]
```

In [199…
```
predictions = model.predict(x_train)
```

In [200…
```
accuracy_score(y_test, predictions)
```

Out[200]:
```
0.25619156521154235
```

In [201…
```
confusion_matrix(y_test, predictions)
```

Out[201]:
```
array([[2515, 3766, 3951, 3438],
       [3783, 5780, 5785, 5332],
       [3875, 5948, 6011, 5422],
       [3423, 5225, 5343, 4738]], dtype=int64)
```

In [202…
```
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.18      0.18      0.18     13670
           1       0.28      0.28      0.28     20680
           2       0.29      0.28      0.28     21256
           3       0.25      0.25      0.25     18729

    accuracy                           0.26     74335
   macro avg       0.25      0.25      0.25     74335
weighted avg       0.26      0.26      0.26     74335
```

Test

In [203…

```
test_predictions = model.predict(x_test)
```

In [204…

```
predictions.shape, test_predictions.shape
```

Out[204]:

```
((74335,), (74335,))
```

In [205…

```
accuracy_score(y_test, test_predictions)
```

Out[205]:

```
0.5804802582901729
```

In [206…

```
confusion_matrix(y_test, test_predictions)
```

Out[206]:

```
array([[ 4868,  1307,  6019,  1476],
       [ 1385, 15029,  2220,  2046],
       [ 6240,  2095, 10540,  2381],
       [ 1588,  1988,  2440, 12713]], dtype=int64)
```

In [207…

```
print(classification_report(y_test, test_predictions))
```

```
              precision    recall  f1-score   support

           0       0.35      0.36      0.35     13670
           1       0.74      0.73      0.73     20680
           2       0.50      0.50      0.50     21256
           3       0.68      0.68      0.68     18729

    accuracy                           0.58     74335
   macro avg       0.57      0.56      0.56     74335
weighted avg       0.58      0.58      0.58     74335
```

In [ ]: