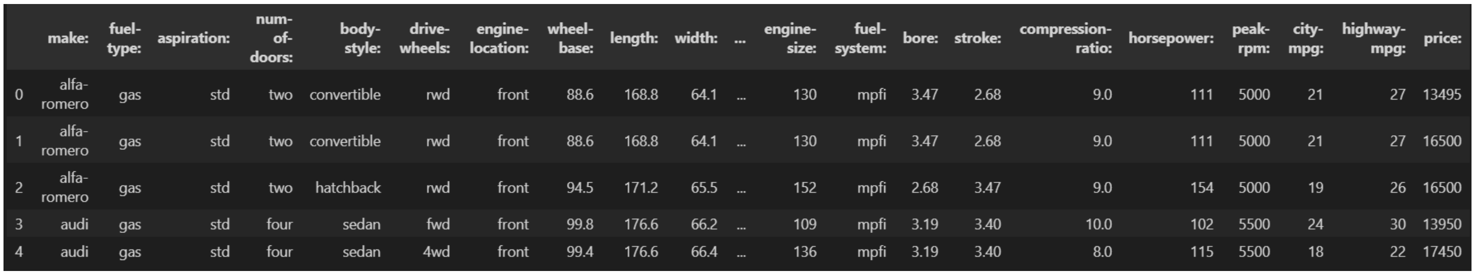
CMPSC 445 – Assignment 4

Gabriel Nulman – gkn5075@psu.edu

Dataset:

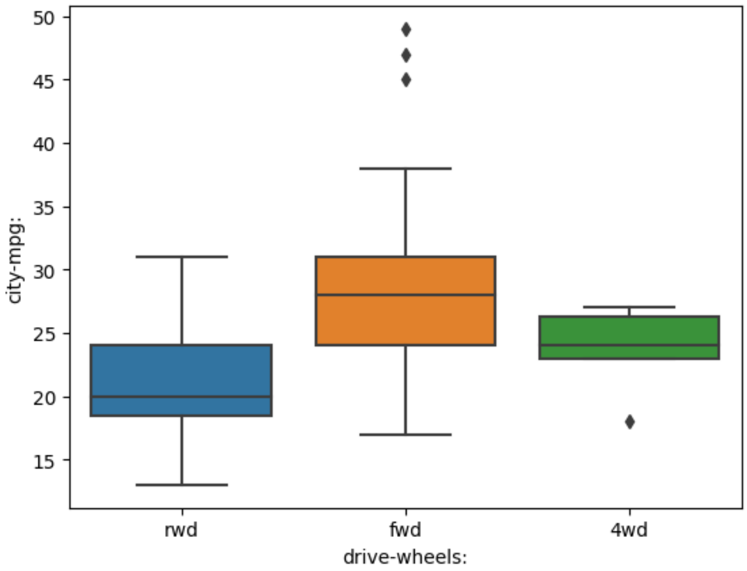
We used the automobiles dataset found at: https://archive.ics.uci.edu/ml/datasets/Automobile

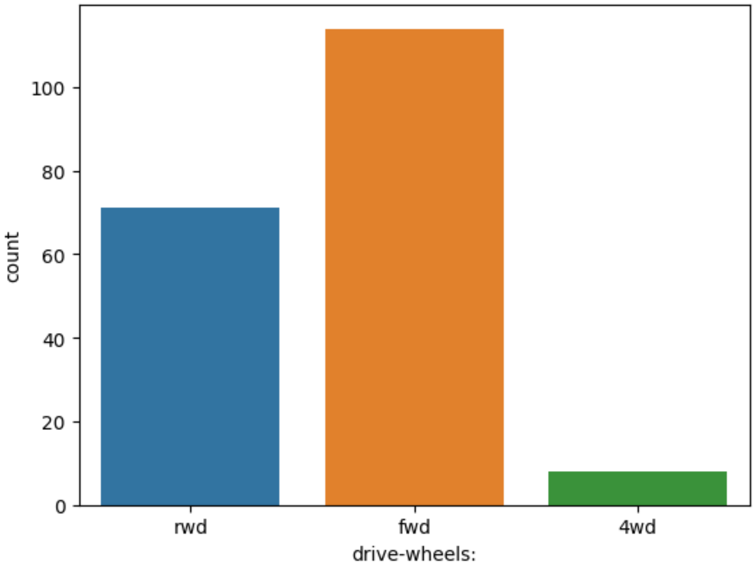
This dataset contains data relevant to certain makes of vehicles such as engine position, MPG, and other related information. I used the KNN algorithm.

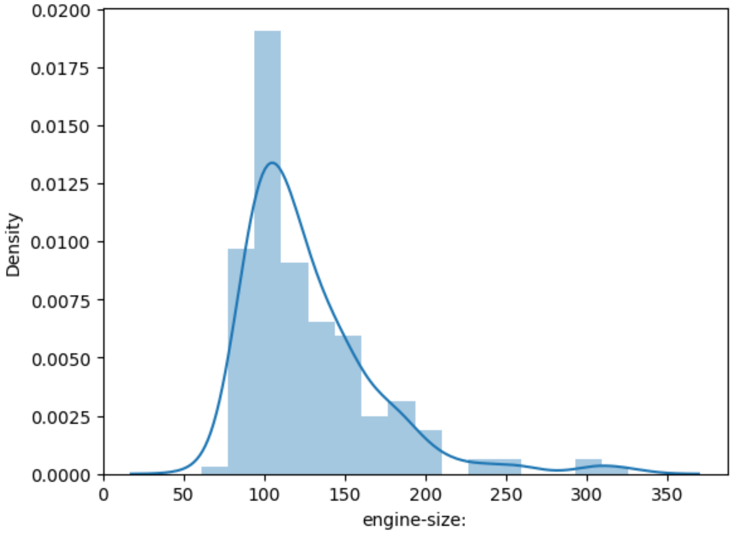


	make:	fuel-type:	aspiration:	num-of-doors:	body-style:	drive-wheels:	engine-location:	wheel-base:	length:	width:	...	engine-size:	fuel-system:	bore:	stroke:	compression-ratio:	horsepower:	peak-rpm:	city-mpg:	highway-mpg:	price:
0	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
1	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
2	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	171.2	65.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500
3	audi	gas	std	four	sedan	fwd	front	99.8	176.6	66.2	...	109	mpfi	3.19	3.40	10.0	102	5500	24	30	13950
4	audi	gas	std	four	sedan	4wd	front	99.4	176.6	66.4	...	136	mpfi	3.19	3.40	8.0	115	5500	18	22	17450

CMPSC 445 – Assignment 4
Gabriel Nulman – gkn5075@psu.edu

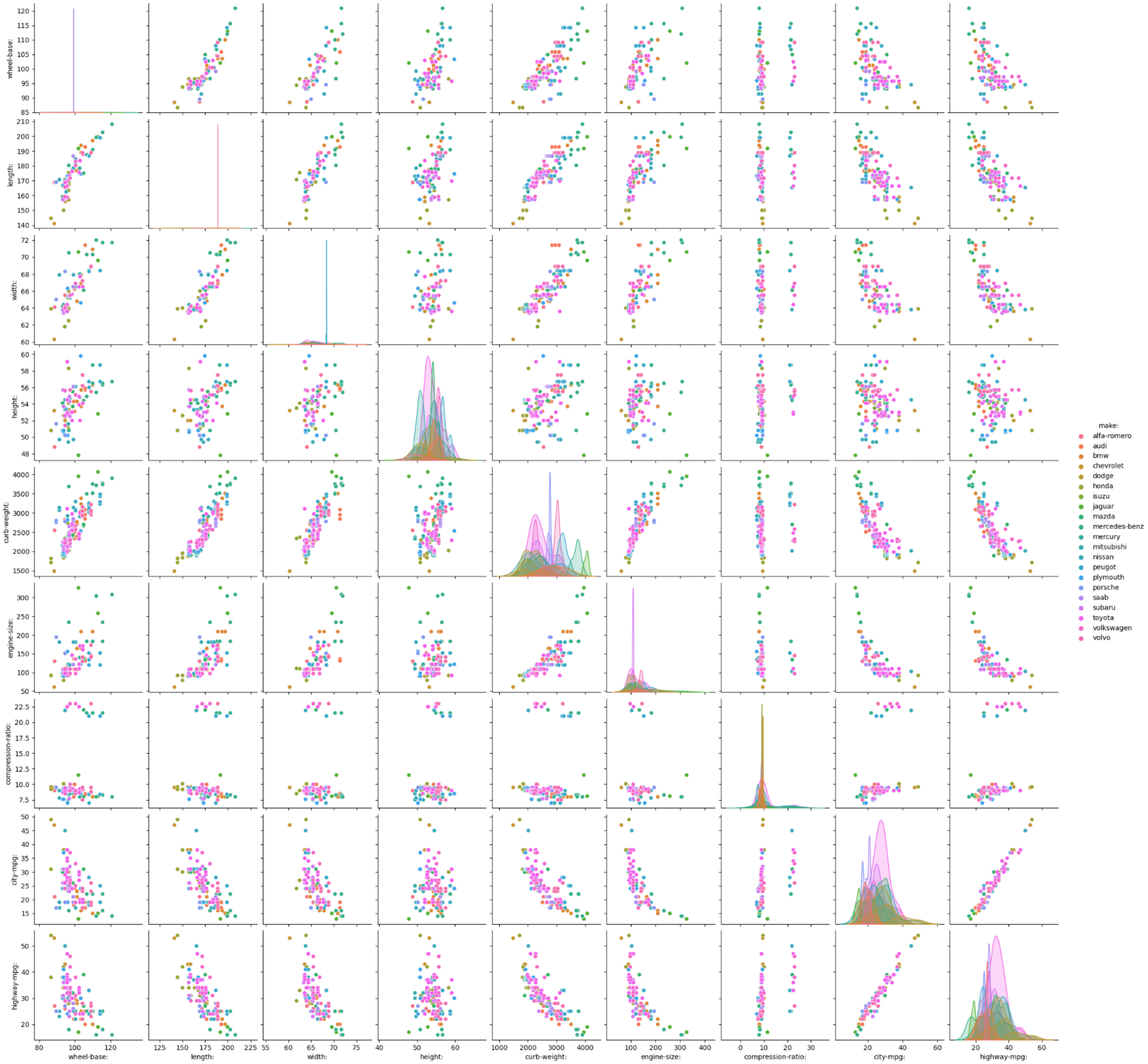




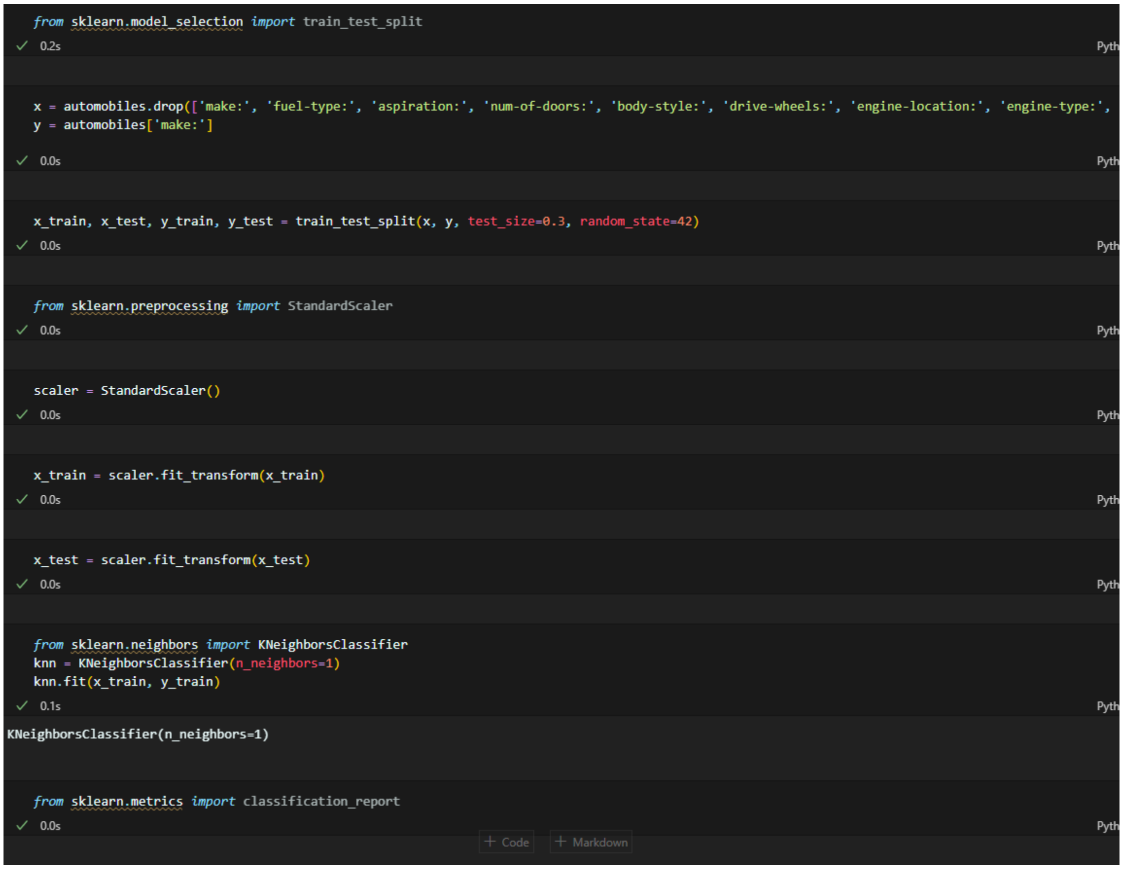


CMPSC 445 – Assignment 4

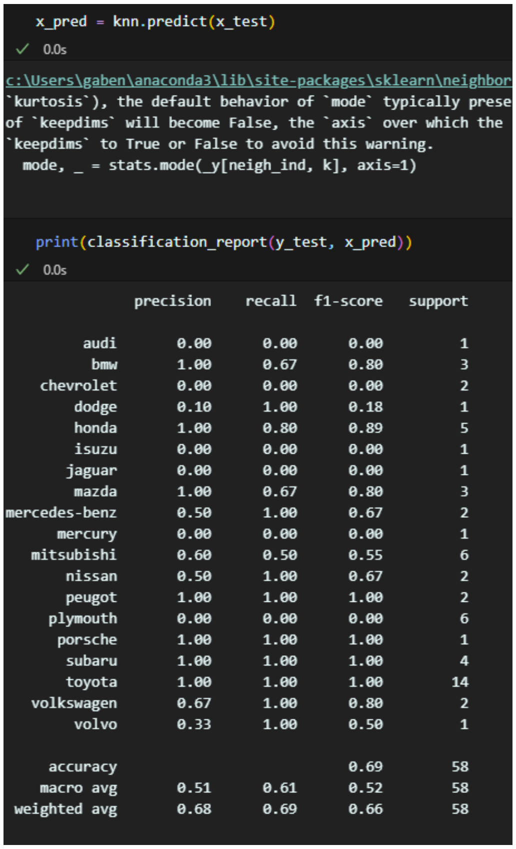
Gabriel Nulman – gkn5075@psu.edu



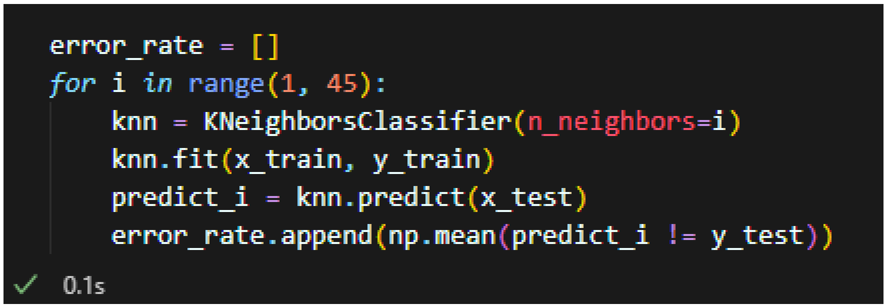
CMPSC 445 – Assignment 4
Gabriel Nulman – gkn5075@psu.edu



```
from sklearn.model_selection import train_test_split
✓ 0.2s

x = automobiles.drop(['make:', 'fuel-type:', 'aspiration:', 'num-of-doors:', 'body-style:', 'drive-wheels:', 'engine-location:', 'engine-type:',
y = automobiles['make:']

✓ 0.0s

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)
✓ 0.0s

from sklearn.preprocessing import StandardScaler
✓ 0.0s

scaler = StandardScaler()
✓ 0.0s

x_train = scaler.fit_transform(x_train)
✓ 0.0s

x_test = scaler.fit_transform(x_test)
✓ 0.0s

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(x_train, y_train)
✓ 0.1s
KNeighborsClassifier(n_neighbors=1)

from sklearn.metrics import classification_report
✓ 0.0s
```

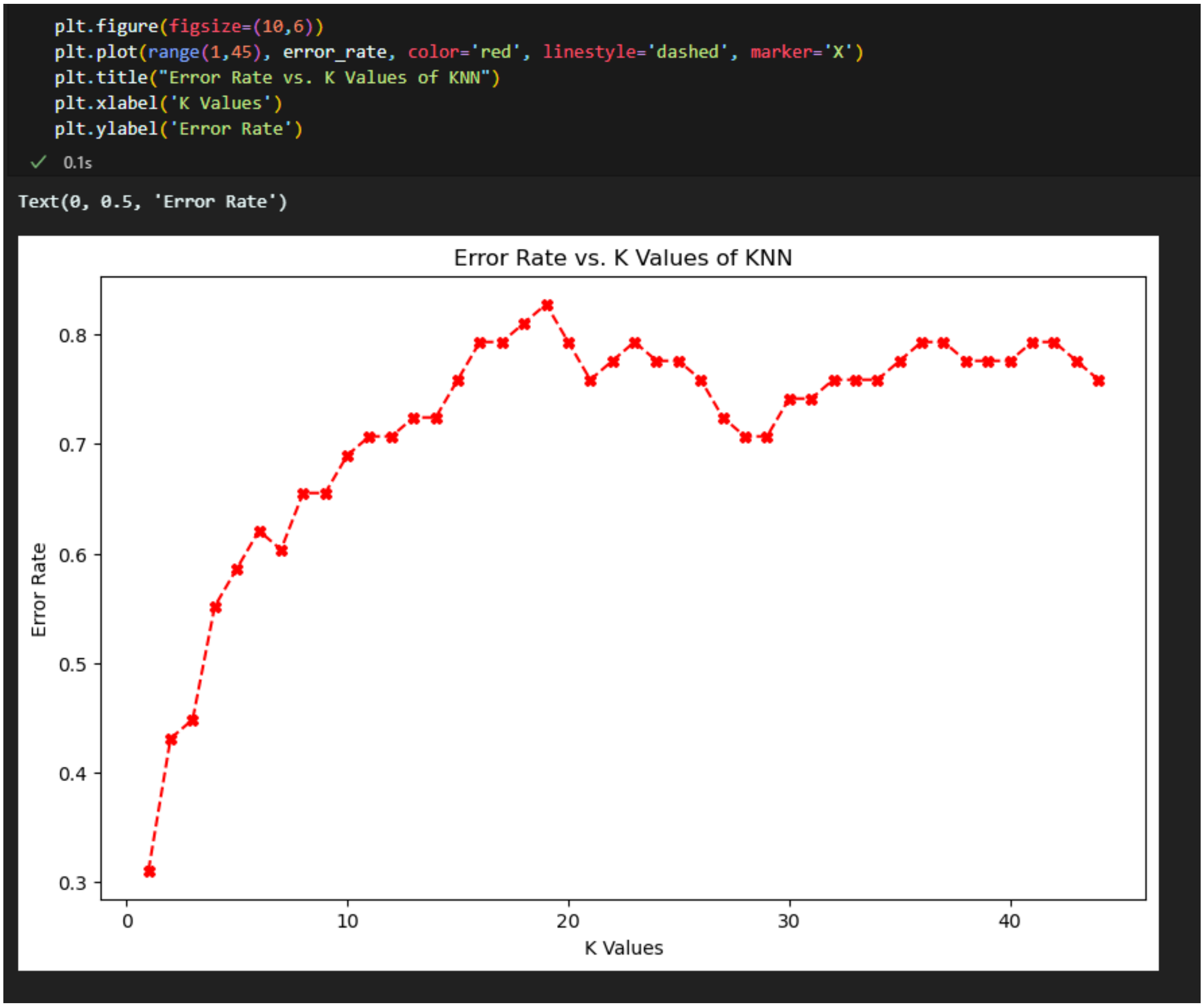


```
x_pred = knn.predict(x_test)
✓ 0.0s
c:\Users\gaben\anaconda3\lib\site-packages\sklearn\neighbor
`kurtosis`), the default behavior of `mode` typically prese
of `keepdims` will become False, the `axis` over which the
`keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)

print(classification_report(y_test, x_pred))
✓ 0.0s
               precision    recall  f1-score   support

         audi       0.00      0.00      0.00         1
          bmw       1.00      0.67      0.80         3
    chevrolet       0.00      0.00      0.00         2
        dodge       0.10      1.00      0.18         1
        honda       1.00      0.80      0.89         5
        isuzu       0.00      0.00      0.00         1
       jaguar       0.00      0.00      0.00         1
        mazda       1.00      0.67      0.80         3
mercedes-benz       0.50      1.00      0.67         2
      mercury       0.00      0.00      0.00         1
   mitsubishi       0.60      0.50      0.55         6
       nissan       0.50      1.00      0.67         2
       peugot       1.00      1.00      1.00         2
     plymouth       0.00      0.00      0.00         6
      porsche       1.00      1.00      1.00         1
       subaru       1.00      1.00      1.00         4
       toyota       1.00      1.00      1.00        14
   volkswagen       0.67      1.00      0.80         2
        volvo       0.33      1.00      0.50         1

     accuracy                           0.69        58
    macro avg       0.51      0.61      0.52        58
 weighted avg       0.68      0.69      0.66        58
```



```
error_rate = []
for i in range(1, 45):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(x_train, y_train)
    predict_i = knn.predict(x_test)
    error_rate.append(np.mean(predict_i != y_test))
```

✓ 0.1s



```
plt.figure(figsize=(10,6))
plt.plot(range(1,45), error_rate, color='red', linestyle='dashed', marker='X')
plt.title("Error Rate vs. K Values of KNN")
plt.xlabel('K Values')
plt.ylabel('Error Rate')
```

✓ 0.1s

```
Text(0, 0.5, 'Error Rate')
```

Conclusion:

To conclude, this very well detailed dataset did not provide great results. Upon realizing when writing this conclusion, this algorithm did not match the dataset as we did not predict any data. If we were to revisit this, we would either use a different dataset or a different algorithm all together.