ISYE 7406 Homework #3

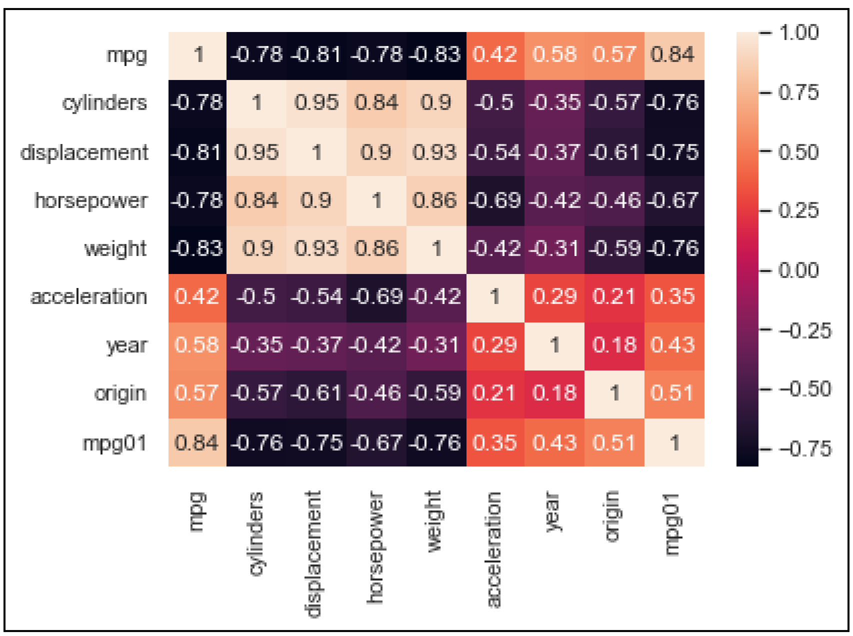
Fall 2022

1. Introduction

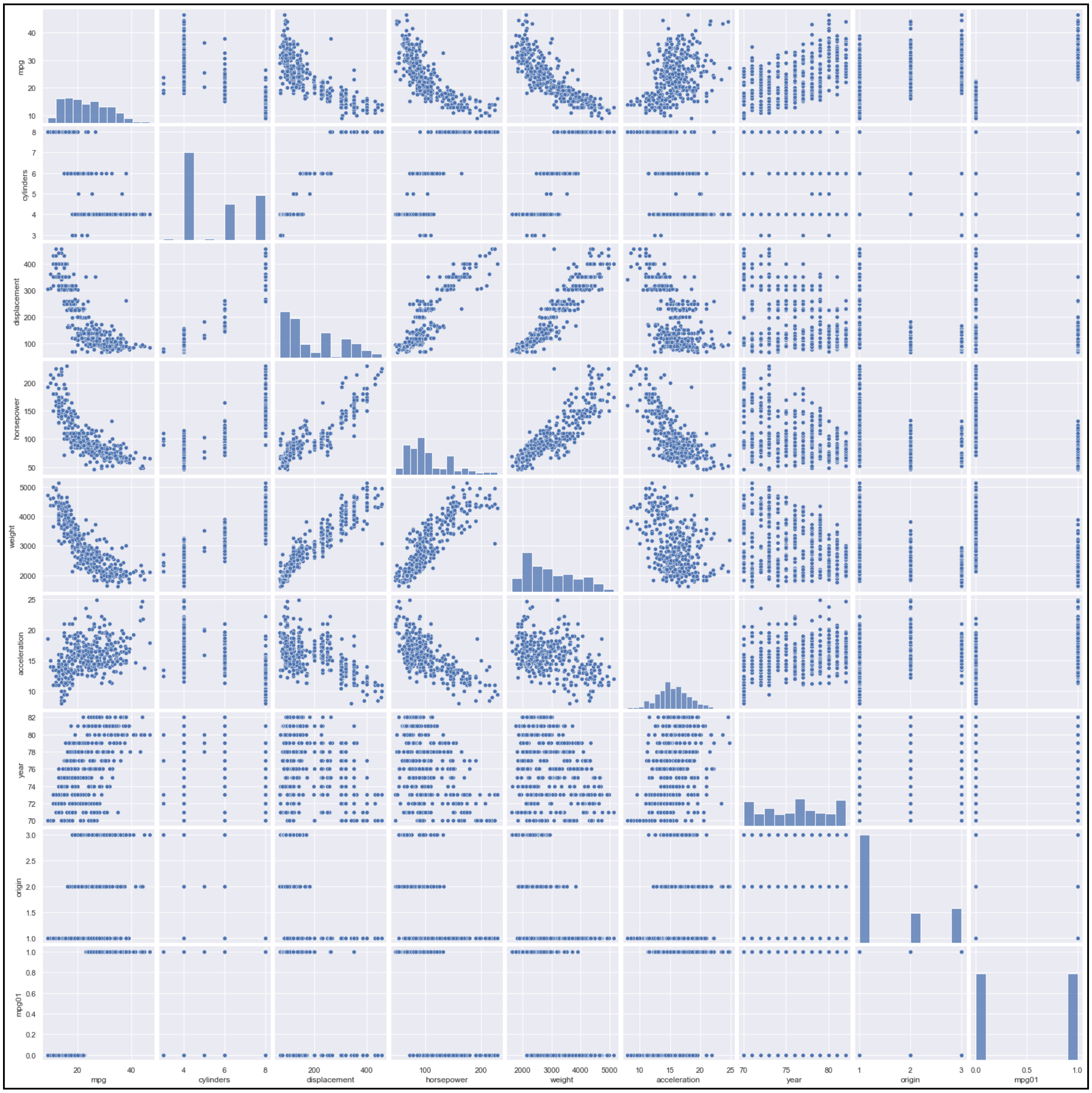
For his homework, we are analyzing the "Auto MPG" dataset with different classification methods. Fortunately, a dataset has been provided to us that has removed rows with missing data and any columns with text so we can perform numerical analysis. There are 392 rows and 8 columns.

2. Exploratory Data Analysis

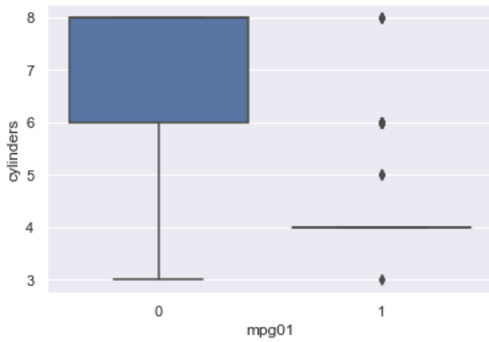
First, we need a column as a response variable in our classification methods. We transform "MPG" into a binary variable by setting its values to "0" or "1" based on whether its value is higher or lower than the median (22.75). Afterwards, we consider taking a look at the correlation between the response and other columns, as well as a pairplot.

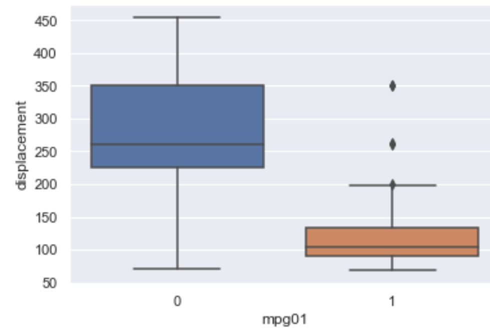


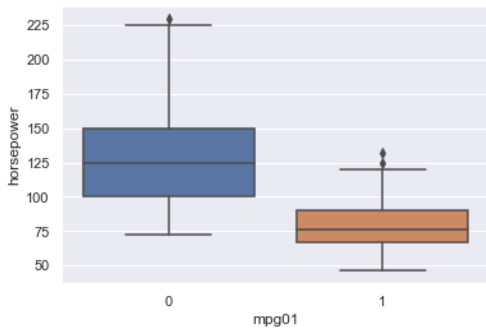
In the figure above, we plot the correlation matrix and see that columns "cylinders", "displacement", "horsepower", "weight", and "origin" all have correlation > 0.5 so we could consider using those variables as our independent variables for classification models.

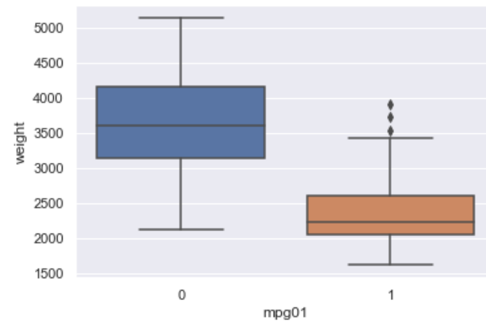


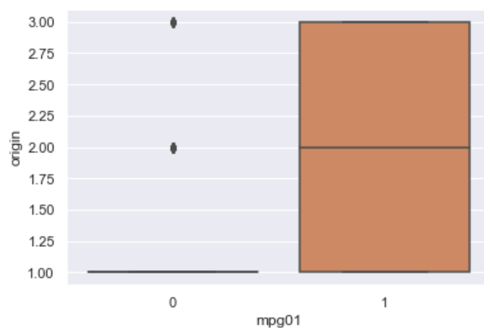
In the pairplot above, we also notice that the most significant columns mentioned above also have a noticeable trending slope with “mpg”. Cylinders, displacement, horsepower, and weight have a downward slope with “mpg”.











For these significant columns, we also plot their boxplots and notice that there are distinct groups for “0” and “1” of the response variable so we can visually classify between the two groups.

We also perform a 80-20 train-test split and the resulting datasets are: “train” with 313 rows and “test” with 79 rows.

3. Methods

Now that we have our independent variables selected and our train/test datasets ready, we can begin comparing different classification methods:

1. Linear Discriminant Analysis
2. Quadratic Discriminant Analysis
3. Naive Bayes

4. Logistic Regression
5. K Nearest Neighbors

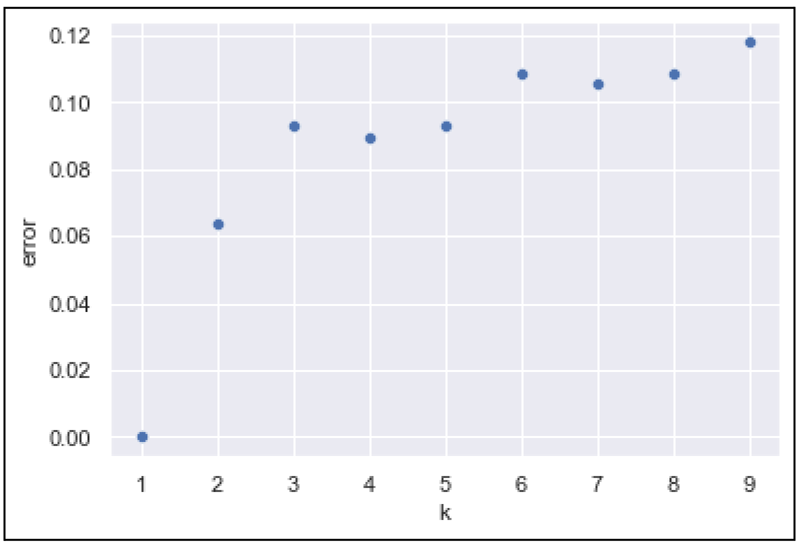
For each method, we will be using its corresponding function in Python's scikit-learn library. We will also evaluate its classification error on the test dataset, as well as performing Monte Carlo cross-validation to compare average error and variance.

4. Results

Test Errors

LDA	QDA	NaiveBayes	LogisticReg	KNN(k=3)
0.0506	0.0632	0.0506	0.1265	0.0886

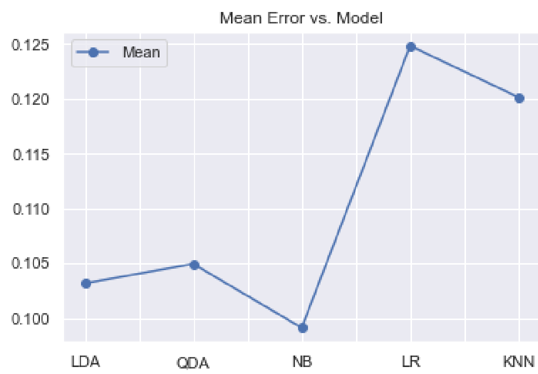
For KNN, we chose k=3 as the number of neighbors to use by comparing train errors for values of k up to 10.

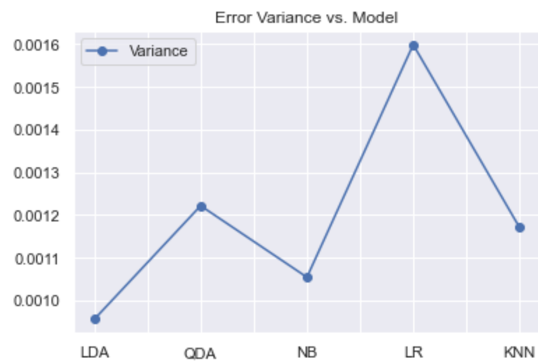


The lowest train error rate is at k=4 (disregarding k=1 because that would provide a severely overfit model), but we select k=3 as the best model since it is the closest odd value (to break ties during voting in KNN).

From evaluating on a single test dataset, we see that LDA and NaiveBayes models performed equally as the best models. We then repeat this process 100 times to perform Monte-Carlo cross validation:

	Mean	Variance
LDA	0.103165	0.000956
QDA	0.104937	0.001222
NB	0.099114	0.001054
LR	0.124810	0.001599
KNN	0.120127	0.001171





From our results above, we notice that the NaiveBayes model performs the best in both single test evaluation and monte-carlo cross validation; it also has the second-lowest variance in error compared to the rest.

5. **Findings**

While we have selected the Naive Bayes model as our best model in this analysis, there are many other factors we are not considering such as comparing various hyper parameters in each classification method. In logistic regression, we could compare various classification threshold levels. For Naive bayes, we could compare different kernel densities. We are also only considering results at an 80-20 split, but could repeat this analysis with a larger or smaller test size.

Appendix: Python Code

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

from sklearn.model_selection import train_test_split
```

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis,
QuadraticDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
df = pd.read_csv("auto.csv")
df['mpg'].median()
df['mpg01'] = df['mpg'].apply(lambda x: 1 if x >= df['mpg'].median() else 0)
df.describe()
sns.heatmap(df.corr(), annot=True)
sns.pairplot(df)
sns.boxplot(df, x='mpg01', y='weight')
sns.boxplot(df, x='mpg01', y='horsepower')
sns.boxplot(df, x='mpg01', y='cylinders')
sns.boxplot(df, x='mpg01', y='displacement')
sns.boxplot(df, x='mpg01', y='origin')
#Remove mpg column from dataset
X = df[['cylinders','displacement','horsepower','weight','origin']]
y = df['mpg01']
#Split dataset into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=7406)
#LDA
lda_model = LinearDiscriminantAnalysis().fit(X_train, y_train)
print("Test Error: ", 1 - lda_model.score(X_test,y_test))
#QDA
qda_model = QuadraticDiscriminantAnalysis().fit(X_train, y_train)
print("Test Error: ", 1 - qda_model.score(X_test,y_test))
#NaiveBayes
nb_model = GaussianNB().fit(X_train, y_train)
print("Test Error: ", 1 - nb_model.score(X_test,y_test))
#LogisticRegression
lr_model = LogisticRegression().fit(X_train, y_train)
print("Test Error: ", 1 - lr_model.score(X_test,y_test))
#KNN
train_errors = []
for k in range(1,10):
    knn_model = KNeighborsClassifier(n_neighbors=k).fit(X_train, y_train)
    train_errors.append(1 - knn_model.score(X_train,y_train))
train_errors

#Use model k=3
knn_model = KNeighborsClassifier(n_neighbors=3).fit(X_train, y_train)
print("Test Error: ", 1 - knn_model.score(X_test,y_test))
knn_train_df = pd.DataFrame({"k":range(1,10), "error":train_errors})
sns.scatterplot(knn_train_df, x="k", y="error")
B = 100 #total loop times
all_test_errors = []

for b in range(B):
    test_errors = []
    #Split train and test dataset based on random state
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=b)

    #1. Train LDA
    lda_model = LinearDiscriminantAnalysis().fit(X_train, y_train)
```

```
    test_errors.append(1 - lda_model.score(X_test,y_test))

    #2. Train QDA
    qda_model = QuadraticDiscriminantAnalysis().fit(X_train, y_train)
    test_errors.append(1 - qda_model.score(X_test,y_test))

    #3. Train NaiveBayes
    nb_model = GaussianNB().fit(X_train, y_train)
    test_errors.append(1 - nb_model.score(X_test,y_test))

    #4. Train logistic regression
    lr_model = LogisticRegression().fit(X_train, y_train)
    test_errors.append(1 - lr_model.score(X_test,y_test))

    #5. Train KNN(k=3)
    knn_model = KNeighborsClassifier(n_neighbors=3).fit(X_train, y_train)
    test_errors.append(1 - knn_model.score(X_test,y_test))

    all_test_errors.append(test_errors)

all_test_errors
all_test_errors = np.array(all_test_errors)
index_names = ['LDA','QDA','NB','LR','KNN']
all_test_err_df = pd.DataFrame({'Mean':np.mean(all_test_errors,axis=0),
                                'Variance':np.var(all_test_errors, axis=0)},
                               index=index_names)
all_test_err_df
all_test_err_df.plot.line(y='Mean',title='Mean Error vs. Model', marker='o')
all_test_err_df.plot.line(y='Variance',title='Error Variance vs. Model',
marker='o')
```