

---

# Do we need more bikes?

## Project in Statistical Machine Learning

---

**Anonymous Author(s)**

Affiliation

Address

email

### Abstract

1 In this project we develop, and study different statistical machine learning models  
2 for predicting whether the number of available bikes at a given hour should be  
3 increased, a project by the District Department of Transportation in Washington  
4 D.C. The training data set consists of 1600 instances of hourly bike rentals, and  
5 a test set of 400 instances. The models for prediction we have used are: *Logistic*  
6 *regression*, *Discriminant methods: LDA, QDA, k- Nearest Neighbour*, and *Tree*  
7 *Based Methods*. We have found that THE MODEL gives best prediction, with  
8 accuracy ??????

## 9 1 Plan

### 10 1.1 From Intro

- 11 (i) Explore and preprocess data
- 12 (ii) try some or all classification methods, which are these?
  - 13 • Logistic Regression
  - 14 • Discriminant analysis: LDA, QDA
  - 15 • K-nearest neighbor
  - 16 • Tree-based methods: classification trees, random forests, bagging
  - 17 • Boosting
- 18 (iii) Which of these are to be "put in production"?

### 19 1.2 From Data analysis task

- 20 • Can any trend be seen comparing different hours, weeks, months?
- 21 • Is there any difference between weekdays and holidays?
- 22 • Is there any trend depending on the weather?

### 23 1.3 From Implementation of methods

24 Each group member should implement one family each, who did what shall be clear!

25 DNNs are encouraged to be implemented, do this if there is time. (DNN is not a thing a group member can claim as their family.)

26 Implement a naive version, let's do: *Always low\_bike\_demand*

#### 28 1.3.1 What to do with each method

- 29 1. Implement the method (each person individually)
- 30 2. Tune hyper-parameters, discuss how this is done (each person individually)
- 31 3. Evaluate with for example cross-validation. Don't use  $E_{k-fold}$  (what is that?) (need to do
- 32 together)
- 33 4. (optional) Think about input features, are all relevant? (together)

34 Before training, unify pre-processing FOR ALL METHODS and choose ONE OR MULTIPLE

35 metrics to evaluate the model. (is it necessary to have the same for all?, is it beneficial?) Examples:

- 36 • accuracy
- 37 • f1-score
- 38 • recall
- 39 • precision

40 Use same test-train split for ALL MODELS

## 41 2 Introduction

42 Statistical machine learning is a subject that aims to build and train algorithms, that analyse large

43 amount of data, and make predictions for the future, which are computed by using established

44 statistical models, and tools from functional analysis. This is a project in supervised, statistical

45 machine learning, where several models were created, and trained, in order to analyse which one of

46 them gives best prediction for the project "Do we need more bikes", where we want to understand,

47 and predict if there is a high, or low demand of city bikes in the public transportation of Washington,

48 a project by the District Department of Transportation in Washington D.C..

49 The data set used for training our models, consist of 15 variables, containing quantitative/qualitative

50 data. We developed several models, and evaluated them with cross-validation, in order to understand

51 which algorithm gives the best prediction.

## 52 3 Theoretical Background

### 53 3.1 Mathematical Overview of the Models

#### 54 3.1.1 Logistic Regression

55 The backbone of logistic regression is linear regression, i.e. finding the least-squares solution to an  
56 equation system

$$X\theta = y \quad (1)$$

57 given by the normal equations

$$X^T X \theta = X^T y \quad (2)$$

58 where  $X$  is the training data matrix,  $\theta$  is the coefficient vector and  $b$  is the training output. The  
59 parameter vector is then used in the sigmoid function:

$$\sigma(z) = \frac{e^z}{1 + e^z} : \mathbb{R} \rightarrow [0, 1], \quad (3)$$

$$z = x^T \theta, \quad (4)$$

60 where  $x$  is the testing input. This gives a statistical interpretation of the input vector. In the case of a  
61 binary True/False classification, the value of the sigmoid function then determines the class.

#### 62 3.1.2 Random forest

63 The random forest method is a based upon decision trees, i.e. dividing the data point into binary  
64 groups based on Gini-impurity, entropy or classification error, Gini being the most common. These  
65 divisions are then used to create a binary tree shown in figure ??Tree) and where thee leaf-nodes  
66 are used to classify the target variables bases on the input. As of itself the disition tree tends to  
67 have unsatisfying results which leads to methodes like random forest and sandbagging that boost its  
68 accuracy. Sandbagging is a way to sampel the data in order to get multiple datasets from the same  
69 data. One then creates a desition-tree for every subset data to then combine them into one model. This  
70 lessens the variance of the model but increases bias. This means that sandbagging can increase false  
71 negatives which in theis aplication makes i nonviable. Random forest on the otherhand is viable, it  
72 creates mutple trees whilse disrecarding random input variable this randomnes decreases overfitting  
73 creating a more robust model.

#### 74 3.1.3 Non-parametric method: k-Nearest Neighbour

75 *k-Nearest Neighbour* ( $k$ -NN) is a distance based method that takes a  $k$  amount of points from the  
76 training data set, called *neighbours*, computes the distance between them, then assumes that the  
77 predicted value  $\hat{y}(x_*)$  follows the trend of the  $k$ - nearest neighbours. Since  $k$ -NN uses the training  
78 data explicitly it is also called a *nonparametric* method.

79 The  $k$ -NN method can be divided into several subcategories, inter alias *classification*  $k$ -NN method,  
80 *regression*  $k$ -NN method. In this project, we are using the classification method, since we are trying  
81 to predict in which of the two classes low, or high demand, the given, and predicted data points  
82 belong.

83 The classification  $k$ -NN algorithm evaluates  $\hat{y}(x_*)$  by computing the most frequently occurring class  
84 among the  $k$  nearest neighbours. Here, we try to identify whether a data point belong to the high  
85 demand-class. Denote  $c$  = high demand class. For simplicity, assume Euclidean ambience. Then

$$\hat{y}(x_*) = \arg \max_c \sum_{n \in \mathbb{N}} \chi_{(y_i=c)},$$

86 where  $y_i$  is the class of the nearest neighbour,  $\chi$  is the characteristic function

$$\chi_{(y_i=c)} = \begin{cases} 1 & \text{if } y_n = c, \\ 0 & \text{otherwise.} \end{cases}$$

87 It is very common to use a weighted sum to predict the next value, i.e.

$$\hat{y}(x_*) = \arg \max_c \sum_{n \in \mathbb{N}} \frac{\chi_{(y_n=c)}}{d(x, x_n)},$$

where  $d$  is the standard Euclidean metric, computing the distance between an input  $x$ , and a neighbour  $x_n$ .

When using this model it is important to choose an optimal  $k$ -value. There are several tests for this, here we implement *uniform weighting*, and *distance weighting*. The first algorithm creates a  $k$ -NN model for each new  $k \in [1, 500]$ , and trains the model with uniform weights, i.e. the contribution of all neighbours is equal. Similarly, the latter trains a  $k$ -NN classifier for each  $k \in [1, 500]$ , with the difference that it uses distance based weighting, i.e. closer neighbours have greater influence. After testing different upper boundaries for  $k$ , the two models gave good results in the interval  $[1, 500]$ , see Figure ?? . From the figures, we can see that the second test gives a better value for  $k$ , since the plot follows smoother trend, in comparison to the uniform weighting test, which makes it easier to identify an optimal  $k$  value ( $k = 120$ ). Moreover, the distance weighting algorithm is providing results for larger values of  $k$ , that is for  $k \in [1, 400)$  before the curve converges, while the uniform weighting algorithm converges earlier, when  $k = 120$ . This means that for large  $k$ , both test algorithms make prediction based on the most common class in the data set, instead of making prediction based on the behaviour of the neighbours. Thus for sufficiently large  $k$ , for any given data point, the model will consider unnecessarily large amount of neighbours, and the prediction will be evaluated to belong to the most frequent class. Since the distance weighting has a larger range of  $k$ -value, it should be more trustworthy.

When  $k = 120$ , the accuracy of the model is 92%.

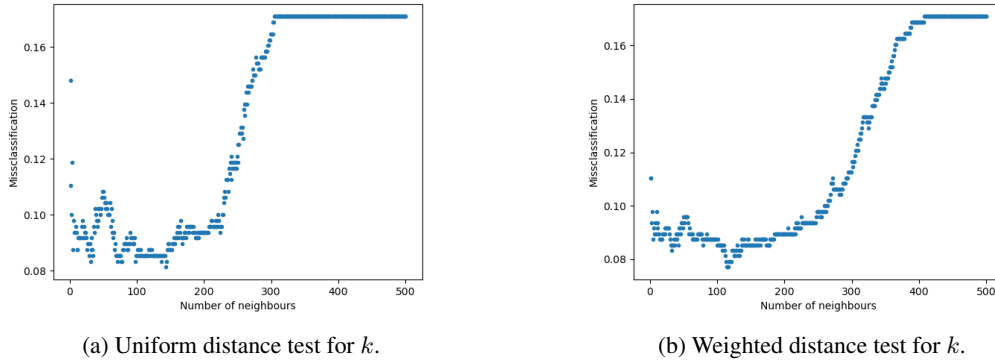


Figure 1: Test for choosing an optimal  $k$ -value.

### 3.1.4 Discriminant analysis: LDA and QDA

Linear Discriminant Analysis is a generative model, which means it is a model that's creating and using a probability distribution  $P(\mathbf{x}, y)$  to create an estimation for the probability  $P(y = m|\mathbf{x})$  using Bayes theorem.

Bayes theorem is:

$$p(y|\mathbf{x}) = \frac{p(y, \mathbf{x})}{p(\mathbf{x})} = \frac{p(y)p(\mathbf{x}|y)}{\int_y p(y, \mathbf{x})}$$

For the discrete version it is obtained:

$$p(y = m|\mathbf{x}) = \frac{p(y = m)p(\mathbf{x}|y = m)}{\sum_{m=1}^M p(y = m)p(\mathbf{x}|y = m)}$$

For this form of the equation to be useful, it is necessary to obtain an accurate estimation of  $p(y = m)$  and  $p(\mathbf{x}|y = m)$  for all classes  $m$ .

In LDA,  $p(y = m)$  is estimated by counting the percentage of data points (in the training data) being in each of the classes and using that percentage as the probability of a data point being in that class.

In mathematical terms:

$$p(y = m) = \frac{1}{n} \sum_{i=1}^n \mathbb{I}\{y_i = m\} = \frac{n_m}{n}$$

118 To estimate the probability distribution  $p(\mathbf{x}|y = m)$ , a multi-dimensional gaussian distribution is  
 119 used:

$$\mathcal{N}(\mathbf{x}|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu)\right)$$

120 Where  $\mathbf{x}$  is the d-dimensional data point,  $\mu$  is the (d-dimensional) mean of the random variable.  $\Sigma$  is  
 121 the symmetric, positive definite covariance matrix defined by:

$$\Sigma = \frac{1}{n - M} \sum_{m=1}^M \sum_{i:y_i=m} (\mathbf{x}_i - \mu_m)(\mathbf{x}_i - \mu_m)^T$$

122 Using these estimations results in an expression for the quantity  $p(y = m|\mathbf{x})\forall m$ . LDA then uses  
 123 maximum likelihood to categorize an input  $\mathbf{x}$  into a class  $m$ .

124  
 125 Quadratic discriminant analysis (QDA) is heavily based of LDA with the sole difference  
 126 being how the covariance matrix  $\Sigma$  is created. In LDA, the covariance matrix is assumed to be the  
 127 same for data in each and every class. In QDA however, the covariance matrix is calculated for each  
 128 class as follows:

$$\Sigma_m = \frac{1}{n_m - 1} \sum_{i:y_i=m} (\mathbf{x}_i - \mu_m)(\mathbf{x}_i - \mu_m)^T$$

129 One thing to note about LDA and QDA is that the use of a multi-variable gaussian distribution  
 130 benefits normally distributed variables. In this project however, there is a dependence on positive  
 131 definite values which are not normally distributed by nature. This is an issue when using QDA since  
 132 in the class of *high\_bike\_demand*, all data points have a snow depth of 0 and has hence no variance.  
 133 This results in this class having a undefined inverse for the covariance matrix. The solution used was  
 134 to exclude this variable from this model.

### 135 3.2 Input Data Modification

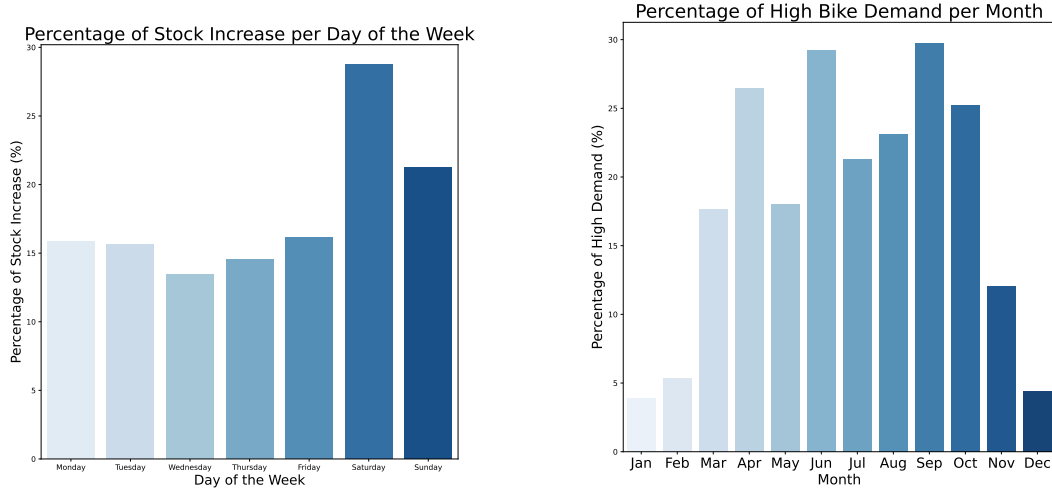
136 By plotting the data and analyzing the .csv file, some observations were made. The different inputs  
 137 were then changed accordingly:

- 138 • *Kept as-is:* weekday, windspeed, visibility, temp
- 139 • *Modified:*
  - 140 – month - split into two inputs, one cosine and one sine part. This make the new inputs
  - 141 linear and can follow the fluctuations of the year. The original input was discarded.
  - 142 – hour\_of\_day - split into three boolean variables: demand\_day, demand\_evening,
  - 143 and demand\_night, reflecting if the time was between 08-14, 15-19 or 20-07 respec-
  - 144 tively. This was done because plotting the data showed three different plateaus of
  - 145 demand for the different time intervals. The original input was discarded.
  - 146 – snowdepth, precip were transformed into booleans, reflecting if it was raining or
  - 147 if there was snow on the ground or not. This was done as there was no times where
  - 148 demand was high when it was raining or when there was snow on the ground.
- 149 • *Removed:* cloudcover, day\_of\_week, snow, dew, holiday, summertime. These were
- 150 removed due to being redundant (e.g. summertime), not showing a clear trend (e.g.
- 151 cloudcover), giving a worse score when used, or all three (e.g. day\_of\_week).

## 152 4 Data Analysis

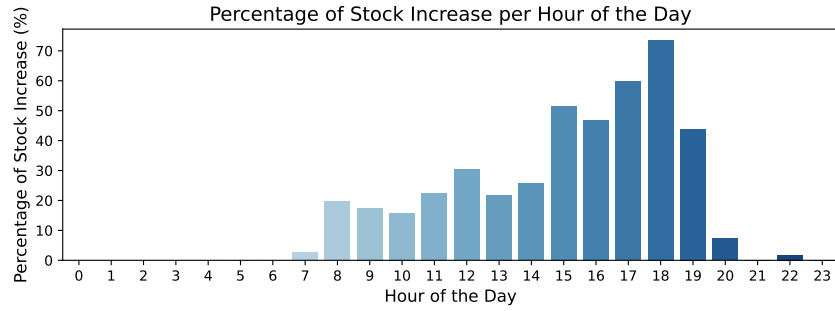
153 In the given data, there are some numerical and categorical features:

- 154 • *Numerical:* temp, dew, humidity, precip, snow, snowdepth, windspeed, cloudcover
- 155 and visibility.
- 156 • *Categorical:* hour\_of\_day, day\_of\_week, month, holiday, weekday, summertime, and
- 157 increase\_stock



(a) Demand per day of week.

(b) Demand per month.



(c) Demand per hour of day.

Figure 2: Bike demand vs. day of week and month.

There are some trends seen in the data when it comes to time and weather. From figure ??, one can see a periodic relationship for the months, where there is a higher demand during the warmer months, loosely following a trigonometric curve. Over the week, the demand is rather stable, with a peak on the weekend, especially Saturdays.

Looking at the weather (figure ??); if there is rain or if there is snow on the ground, there is close to always low demand. Cloudcover did not make a big impact, which is also intuitive, as a cloudy day does not make biking more difficult. Dew point also does not have a clear trend, while humidity however has a clear trend downwards as the humidity increases. Temperature had a more clear impact, where more people wanted to bike the warmer it got.

The overall trend is that about one eighth of observations correspond to a high bike demand. During the night, or in bad weather, the demand is (intuitively) low. But during rush hour (figure ??), the demand is very high, and should probably be increased in order to minimize excessive CO<sub>2</sub> emissions.

## 5 Result

The method used to evaluate the different models where simply chosen to be the accuracy defined by:

$$\text{accuracy} = \frac{n_{\text{correct}}}{n_{\text{tot}}}$$

The different models were tested and the accuracy where:

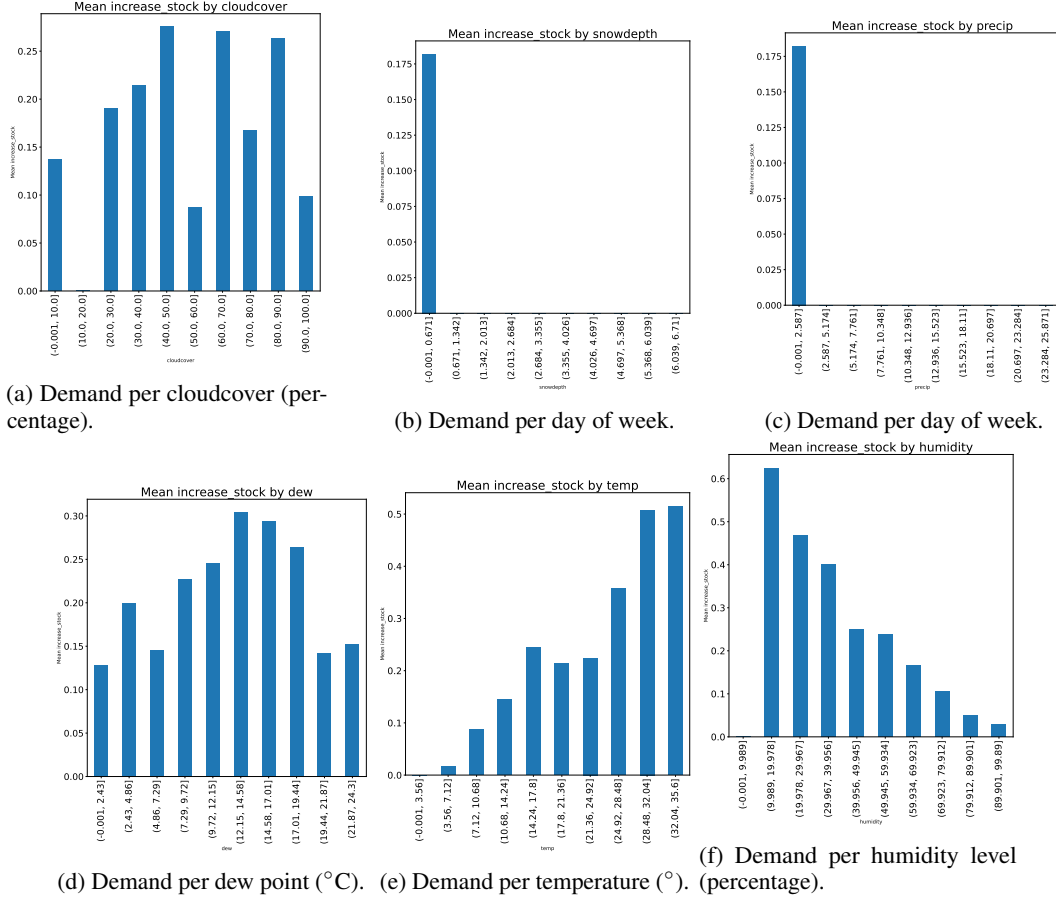


Figure 3: Bike demand vs. various weather parameters.

Accuracy of the models	
Model	Accuracy
LDA	86%
QDA	86%
Random forrest	91%

Here you can clearly see random forset is the best clasifier and is the model that the group would use in production. This because it has the highest accuracy score and f1 scora for high<sub>d</sub>emand. Thesecondpartbeingthemostimoprtantcriteriameaningconsideringinthisaplicationfalsenegativesare

## 174 A Appendix

```

175 1 import pandas as pd
176 2 import numpy as np
177 3 from sklearn.model_selection import train_test_split
178 4 from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
179 5 from sklearn.linear_model import LogisticRegression
180 6 from sklearn.metrics import accuracy_score
181 7 from sklearn.metrics import classification_report
182 8
183 9 df = pd.read_csv('training_data_vt2025.csv')
184 10
185 11 # modify the month to represent the periodicity that is observed in
186 12 data.
187 13 df['month_cos'] = np.cos(df['month']*2*np.pi/12)
188 14 df['month_sin'] = np.sin(df['month']*2*np.pi/12)
189 15
190 16 # time of day, replaced with 3 bool values: is_night, is_day and
191 17 is_evening,
192 18 # adding the new categories back in the end.
193 19 def categorize_demand(hour):
194 20     if 20 <= hour or 7 >= hour:
195 21         return 'night'
196 22     elif 8 <= hour <= 14:
197 23         return 'day'
198 24     elif 15 <= hour <= 19:
199 25         return 'evening'
200 26
201 27 df['time_of_day'] = df['hour_of_day'].apply(categorize_demand)
202 28 df_dummies = pd.get_dummies(df['time_of_day'], prefix='is', drop_first
203 29 =False)
204 30 df = pd.concat([df, df_dummies], axis=1)
205 31
206 32 # Create bool of snowdepth and percipitation
207 33 df['snowdepth_bool'] = df['snowdepth'].replace(0, False).astype(bool)
208 34 df['precip_bool'] = df['precip'].replace(0, False).astype(bool)
209 35
210 36 # Seperate training data from target
211 37 X=df[['#holiday',
212 38     'weekday',
213 39     '#summertime',
214 40     'temp',
215 41     '#dew',
216 42     '#humidity',
217 43     '#visibility',
218 44     '#windspeed',
219 45     '#month',
220 46     'month_cos',
221 47     'month_sin',
222 48     '#hour_of_day',
223 49     'is_day',
224 50     'is_evening',
225 51     'is_night',
226 52     '#hour_cos',
227 53     '#hour_sin',
228 54     'snowdepth_bool',
229 55     'precip_bool'
230 56 ]]
231 57
232 58 y=df['increase_stock']
233 59
234 60 # Split dataset into training and test sets
235 61 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
236 62 =0.2, random_state=42)
237 63

```



```

23860 # Apply Linear Discriminant Analysis (LDA)
23961 lda = LinearDiscriminantAnalysis(n_components=1)
24062 X_train_lda = lda.fit_transform(X_train, y_train)
24163 X_test_lda = lda.transform(X_test)
24264
24365 # Train a classifier (Logistic Regression)
24466 clf = LogisticRegression()
24567 clf.fit(X_train_lda, y_train)
24668
24769 # Make predictions
24870 y_pred = clf.predict(X_test_lda)
24971
25072 # Evaluate accuracy
25173 accuracy = accuracy_score(y_test, y_pred)
25274 print(f"Model Accuracy: {accuracy:.2f}")
25375
25476 print(classification_report(y_test, y_pred))

```

Listing 1: Code for LDA

```

255 1 import pandas as pd
256 2 import numpy as np
257 3 from sklearn.model_selection import train_test_split
258 4 from sklearn.discriminant_analysis import
259     QuadraticDiscriminantAnalysis
260 5 from sklearn.metrics import accuracy_score
261 6 from sklearn.metrics import classification_report
262 7
263 8 df = pd.read_csv('training_data_vt2025.csv')
264 9
26510 # modify the month to represent the periodicity that is observed in
266     data.
26711 df['month_cos'] = np.cos(df['month']*2*np.pi/12)
26812 df['month_sin'] = np.sin(df['month']*2*np.pi/12)
26913
27014 # time of day, replaced with 3 bool values: is_night, is_day and
271     is_evening,
27215 # adding the new categories back in the end.
27316 def categorize_demand(hour):
27417     if 20 <= hour or 7 >= hour:
27518         return 'night'
27619     elif 8 <= hour <= 14:
27720         return 'day'
27821     elif 15 <= hour <= 19:
27922         return 'evening'
28023
28124 df['time_of_day'] = df['hour_of_day'].apply(categorize_demand)
28225 df_dummies = pd.get_dummies(df['time_of_day'], prefix='is', drop_first
283     =False)
28426 df = pd.concat([df, df_dummies], axis=1)
28527
28628 # Create bool of snowdepth and percipitation
28729 df['snowdepth_bool'] = df['snowdepth'].where(df['snowdepth'] == 0, 1)
28830 df['precip_bool'] = df['precip'].where(df['precip'] == 0, 1)
28931
29032 # Seperate training data from target
29133 X=df[['#holiday',
29234     'weekday',
29335     '#summertime',
29436     'temp',
29537     '#dew',
29638     '#humidity',
29739     '#visibility',
29840     '#windspeed',
29941     '#month',

```

```

30042         'month_cos',
30143         'month_sin',
30244         #'hour_of_day',
30345         'is_day',
30446         'is_evening',
30547         'is_night',
30648         #'snowdepth_bool',
30749         'precip_bool'
30850     ]]
30951
31052 y=df['increase_stock']
31153
31254 # Split dataset into training and test sets
31355 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
314         =0.2, random_state=42)
31556
31657 # Apply Quadratic Discriminant Analysis (QDA)
31758 qda = QuadraticDiscriminantAnalysis()
31859 X_train_lda = qda.fit(X_train, y_train)
31960
32061 # Make predictions
32162 y_pred = qda.predict(X_test)
32263
32364 # Evaluate accuracy
32465 accuracy = accuracy_score(y_test, y_pred)
32566 print(f"Model Accuracy: {accuracy:.2f}")
32667
32768 print(classification_report(y_test, y_pred))

```

Listing 2: Code for QDA