Which bridge is a better choice? 520 or i-90?

CS 5001 Discrete Structures | Final Project Report

Team: Peihsuan Lin, Yi-chia Chu, Huiru Yang

Abstract: This report aims to analyze the traffic data of the 520 and I 90 bridge in the Seattle area and give proper recommendations to commuters. We first apply conditional probability theory to the question, and then we link our question to Bayes' theorem. We then compare the Bayes model with other random forest models, and give proper recommendations.

1. Introduction

1.1. The Question “Which bridge is a better choice: SR-520 or I-90?”

Every day, commuters in the Seattle area face the dilemma of deciding which bridge to take between Seattle and Bellevue. While some navigation apps suggest taking SR-520 Bridge, others recommend taking I-90 Bridge instead. Our goal is to apply principles of Bayes' theorem to analyze traffic flow data and other pertinent factors to determine the most suitable bridge. Additionally, we will compare the results with other models to provide a more comprehensive conclusion and suitable recommendations.

1.2. Relations with Bayes' theorem

To answer the question, we initially apply the definition of conditional probability, which signifies the probability of an event (or multiple events) given the occurrence of another event, e.g. P(A given B) or P(A | B). Then we got:

- *P(I90 is selected | gas price is at x dollar)*
- *P(SR520 is selected | gas price is at x dollar)*
- *P(I90 is selected | congestion going on)*
- *P(SR520 is selected | congestion going on)*

and so on….

In general, P(A|B) denotes the posterior probability, while P(A) is referred to as the prior probability. In order to get the results of our P(A|B)s, we then apply Bayes' theorem, which we have learned in class:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B|A) * P(A) + P(B|\neg A) * P(\neg A)}$$

Alternatively, Bayes Theorem can also be expressed as:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

By applying our P(A|B)s, we can get:

- *<u>P(I90 is selected | gas price is at x number)</u> = P(gas price is at x number | I90 is selected) * P(I90 is selected) / P(gas price is at x number)*

- *P(SR520 is selected | gas price is at x number) = P(gas price is at x number | SR520 is selected) * P(SR520 is selected) / P(gas price is at x number)*
- *P(I90 is selected | there has construction going on) = P(there has construction going on | I90 is selected) * P(I90 is selected) / P(there has construction going on)*
- *P(SR520 is selected | there has construction going on) = P(there has construction going on | SR520 is selected) * P(SR520 is selected) / P(there has construction going on)*

and so on….

In conclusion, we can have conventional equations:

- *P(I90 is selected | variable x) = P(variable x | I90 is selected) * P(I90 is selected) / P(variable x)*
- *P(SR520 is selected | a variable) = P(variable x | SR520 is selected) * P(SR520 is selected) / P(variable x)*

Then we can implement a simple calculation through Python:

```python
# calculate P(A|B) given P(A), P(B|A), P(B|not A)
# A1: I90 is selected
# A2: SR520 is selected
# B: a variable, b variable.....etc
def bayes_theorem(p_a, p_b_given_a, p_b_given_not_a):
 # calculate P(not A)
 not_a = 1 - p_a
 # calculate P(B)
 p_b = p_b_given_a * p_a + p_b_given_not_a * not_a
 # calculate P(A|B)
 p_a_given_b = (p_b_given_a * p_a) / p_b
 return p_a_given_b
```

1.3. Process

Once we connected our question with Bayes' Theorem, we considered the number of variables that we need to take into consideration. Numerous factors would impact while determining the smoother route. However, given the short timeframe, our scope is narrowed down to the East and West regions of Washington and our primary focus will be on utilizing the following independent variables:

1. *Traffic Volume*
2. *Toll Fee*
3. *Regular Gas Price in Washington State*
4. *Travel Time*

2. Analysis

2.1. Methodology

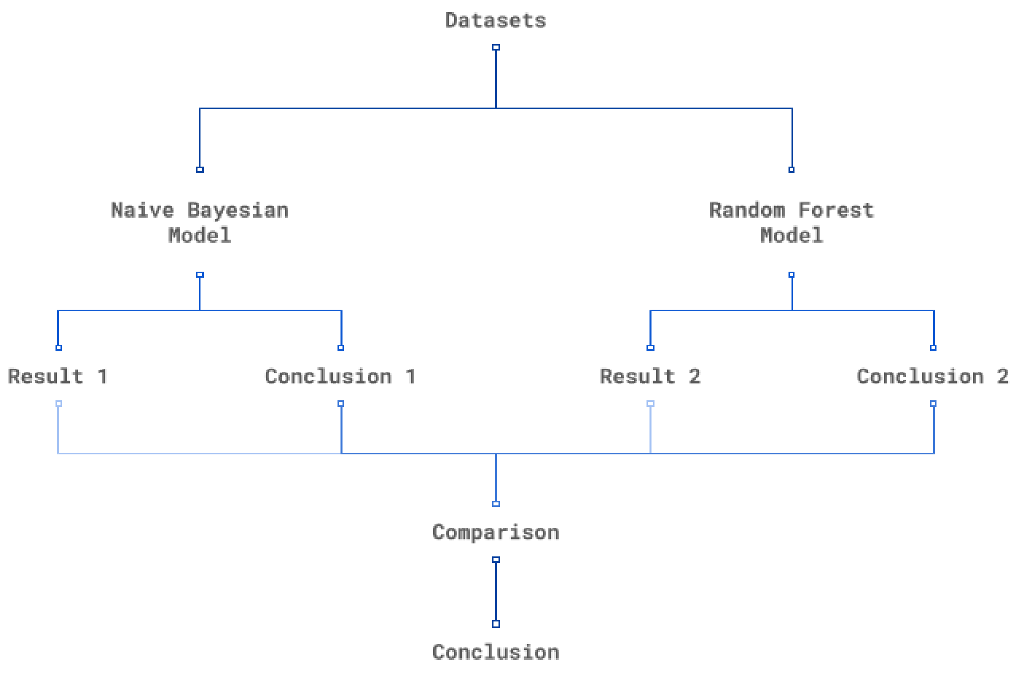


Figure 1: Methodology

Inputting each independent variable to Bayes Theorem would make the way of thinking our question as binary classification. As a result, the direct application of Bayes Theorem will become intractable, especially in our case, where there are several numbers of variables.

Therefore, we will utilized Naive Bayes Classifier to categorize whether an individual should opt for SR520 or i90 based on measured features, this entails multiplying P(data|class) for each variable collectively:

$$P(class \mid X1, X2, \ldots, Xn) = P(X1|class) * P(X2|class) * \ldots * P(Xn|class) * P(class) / P(data)$$

In addition, we will also apply Random Forest Classifier, which is among the most widely used models given its effectiveness in complex scenarios. We will compare the results between the two and select the most suitable one.

2.2. Data collection

a. Monthly Traffic Volume

Monthly traffic data for 520 was collected from the Washington State Department of Transportation (WSDOT) website under the section Tolling reports & policy: Toll Traffic and Revenue, time period is from 2012 to 2022 (refer to Appendix 4.1). We took the reported transaction number as monthly traffic volume since it represents the number of vehicles passing through 520 bridge in each month.

Monthly Traffic Volume(520) = Reported Transactions

Monthly traffic data for I90 was collected from the Traffic Count Database System (TCDS) operated by Washington State Department of Transportation (WSDOT), time period is from 2012 to 2022. The original dataset maps average hourly traffic volume on a monthly basis (refer to Appendix 4.1). For us:

Monthly Traffic Volume(I90) = Total Count(monthly average hourly traffic volume)

b. Monthly Regular Gas Price

Washington's monthly regular gas price was collected from Reno Gazette Journal’s database. In the database, for Washington Regular Gasoline Price History, it contains data from 2023 April to 2003, we used data from 2012 to 2022, in the original data, it recorded 4 times each month, which is the average weekly price (refer to Appendix 4.1). For us:

Monthly Regular Gas Price(WA) = Average(Total Count(weekly average reg gasoline price))

c. Toll Fee

The toll fee data we used was calculated using the same data from the WSDOT website under the section Tolling reports & policy: Toll Traffic and Revenue (refer to Appendix 4.1).

Average Monthly Toll Fee = Reported values / Reported Transactions

d. Travel Time

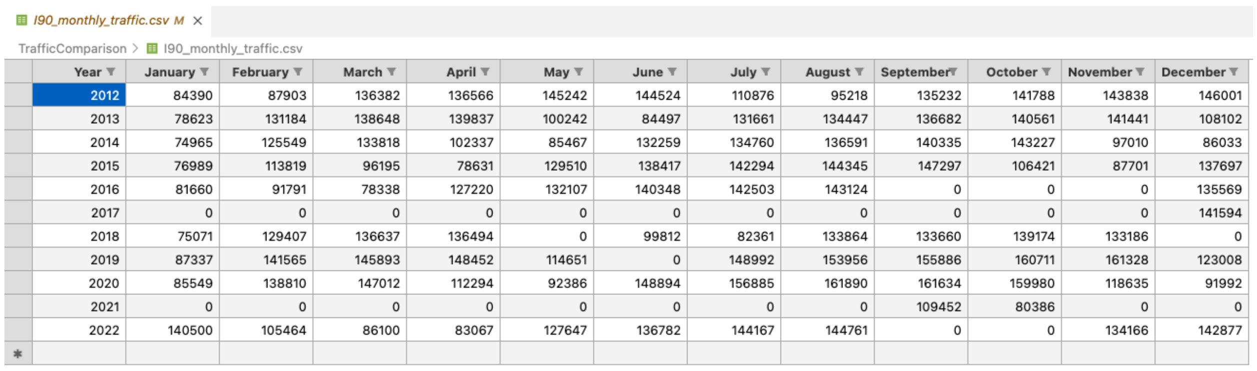
Both I90 and SR520’s travel time data were collected from Washington State Department of Transportation (WSDOT) website under the section Multimodal mobility dashboard’s Commute times.

We use the average morning and evening peak commute times for weekday trips on I-90 and SR520 between Seattle and Bellevue. Time ranges from 2018 to 2021, but we decided to use the latest data, from the year 2021 (refer to Appendix 4.1).

2.3. Dataset Retrieving

a. Monthly Traffic Volume

For I-90, we download .xlsx files from the website, then we implement a python script to automate the process of retrieving the desired data and writing it in a .csv file (refer to Appendix 4.2 - 1), with Year as index, and Month as value columns (12 rows x 13 columns):



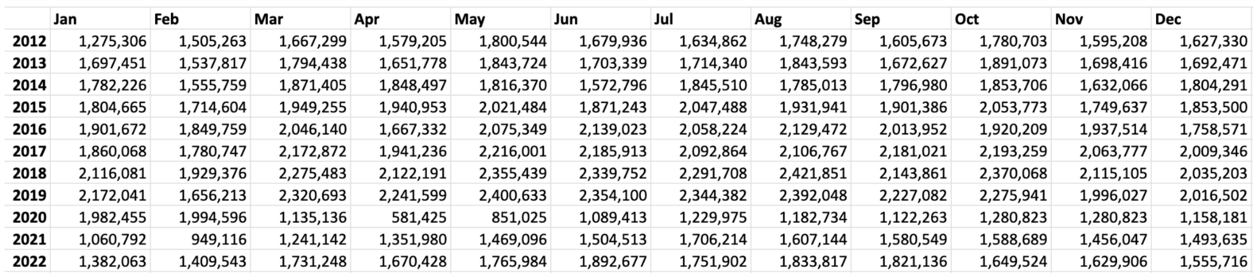
I90_monthly_traffic.csv M ×

TrafficComparison > I90_monthly_traffic.csv

Year	January	February	March	April	May	June	July	August	September	October	November	December
2012	84390	87903	136382	136566	145242	144524	110876	95218	135232	141788	143838	146001
2013	78623	131184	138648	139837	100242	84497	131661	134447	136682	140561	141441	108102
2014	74965	125549	133818	102337	85467	132259	134760	136591	140335	143227	97010	86033
2015	76989	113819	96195	78631	129510	138417	142294	144345	147297	106421	87701	137697
2016	81660	91791	78338	127220	132107	140348	142503	143124	0	0	0	135569
2017	0	0	0	0	0	0	0	0	0	0	0	141594
2018	75071	129407	136637	136494	0	99812	82361	133864	133660	139174	133186	0
2019	87337	141565	145893	148452	114651	0	148992	153956	155886	160711	161328	123008
2020	85549	138810	147012	112294	92386	148894	156885	161890	161634	159980	118635	91992
2021	0	0	0	0	0	0	0	0	109452	80386	0	0
2022	140500	105464	86100	83067	127647	136782	144167	144761	0	0	134166	142877

Figure 2: I-90 Monthly Traffic Volume from 2012-2022

For SR 520, since the raw data is well organized, we migrated them into the previous one.



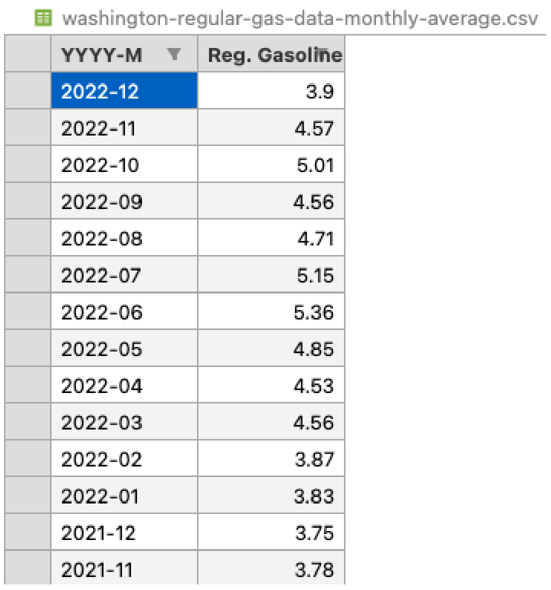
	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2012	1,275,306	1,505,263	1,667,299	1,579,205	1,800,544	1,679,936	1,634,862	1,748,279	1,605,673	1,780,703	1,595,208	1,627,330
2013	1,697,451	1,537,817	1,794,438	1,651,778	1,843,724	1,703,339	1,714,340	1,843,593	1,672,627	1,891,073	1,698,416	1,692,471
2014	1,782,226	1,555,759	1,871,405	1,848,497	1,816,370	1,572,796	1,845,510	1,785,013	1,796,980	1,853,706	1,632,066	1,804,291
2015	1,804,665	1,714,604	1,949,255	1,940,953	2,021,484	1,871,243	2,047,488	1,931,941	1,901,386	2,053,773	1,749,637	1,853,500
2016	1,901,672	1,849,759	2,046,140	1,667,332	2,075,349	2,139,023	2,058,224	2,129,472	2,013,952	1,920,209	1,937,514	1,758,571
2017	1,860,068	1,780,747	2,172,872	1,941,236	2,216,001	2,185,913	2,092,864	2,106,767	2,181,021	2,193,259	2,063,777	2,009,346
2018	2,116,081	1,929,376	2,275,483	2,122,191	2,355,439	2,339,752	2,291,708	2,421,851	2,143,861	2,370,068	2,115,105	2,035,203
2019	2,172,041	1,656,213	2,320,693	2,241,599	2,400,633	2,354,100	2,344,382	2,392,048	2,227,082	2,275,941	1,996,027	2,016,502
2020	1,982,455	1,994,596	1,135,136	581,425	851,025	1,089,413	1,229,975	1,182,734	1,122,263	1,280,823	1,280,823	1,158,181
2021	1,060,792	949,116	1,241,142	1,351,980	1,469,096	1,504,513	1,706,214	1,607,144	1,580,549	1,588,689	1,456,047	1,493,635
2022	1,382,063	1,409,543	1,731,248	1,670,428	1,765,984	1,892,677	1,751,902	1,833,817	1,821,136	1,649,524	1,629,906	1,555,716

Figure 3: SR 520 Monthly Traffic Volume from 2012-2022

b. Monthly Regular Gas Price

The original data was displayed on the website without access for downloading. We also implemented a python script to automate the process of retrieving the desired data and writing it in a .csv file (refer to Appendix 4.2 - 2).

Finally we got the monthly regular gas price data in .csv file with Year-month as index and corresponding reg. gas price as value column (237 rows x 2 columns):

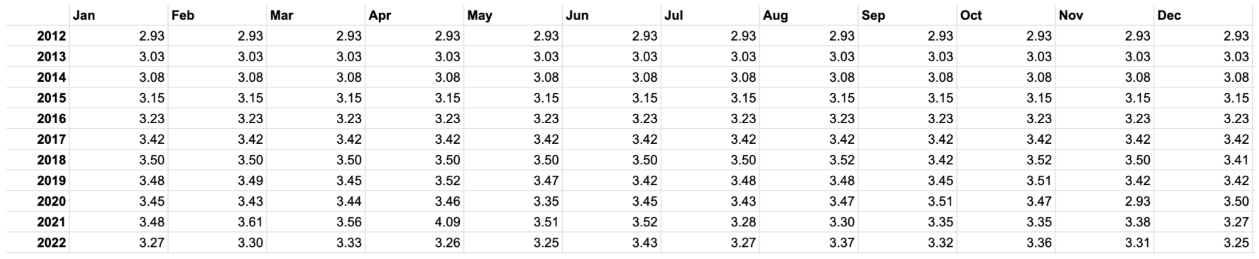

washington-regular-gas-data-monthly-average.csv

YYYY-M	Reg. Gasoline
2022-12	3.9
2022-11	4.57
2022-10	5.01
2022-09	4.56
2022-08	4.71
2022-07	5.15
2022-06	5.36
2022-05	4.85
2022-04	4.53
2022-03	4.56
2022-02	3.87
2022-01	3.83
2021-12	3.75
2021-11	3.78

Figure 4: Monthly Regular Gas Price from 2021-2022

c. Toll Fee

We merge our SR520 traffic volume data and the data of reported monthly revenue to an excel file to apply the formula for calculating the toll fee: *Average Monthly Toll Fee = Reported values / Reported Transactions*, with Year as index, and Month as value columns. For toll fee data from 2012 to 2018 July, since we did not have monthly reported revenue for this period, we used average annual toll fee instead. As for i90, since the bridge has no toll fee, we simply migrate the two together.

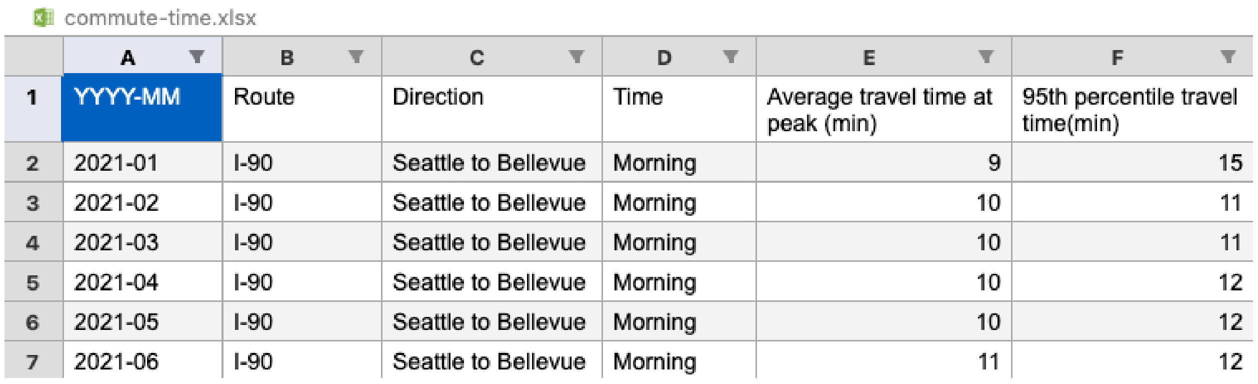


	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2012	2.93	2.93	2.93	2.93	2.93	2.93	2.93	2.93	2.93	2.93	2.93	2.93
2013	3.03	3.03	3.03	3.03	3.03	3.03	3.03	3.03	3.03	3.03	3.03	3.03
2014	3.08	3.08	3.08	3.08	3.08	3.08	3.08	3.08	3.08	3.08	3.08	3.08
2015	3.15	3.15	3.15	3.15	3.15	3.15	3.15	3.15	3.15	3.15	3.15	3.15
2016	3.23	3.23	3.23	3.23	3.23	3.23	3.23	3.23	3.23	3.23	3.23	3.23
2017	3.42	3.42	3.42	3.42	3.42	3.42	3.42	3.42	3.42	3.42	3.42	3.42
2018	3.50	3.50	3.50	3.50	3.50	3.50	3.50	3.52	3.42	3.52	3.50	3.41
2019	3.48	3.49	3.45	3.52	3.47	3.42	3.48	3.48	3.45	3.51	3.42	3.42
2020	3.45	3.43	3.44	3.46	3.35	3.45	3.43	3.47	3.51	3.47	2.93	3.50
2021	3.48	3.61	3.56	4.09	3.51	3.52	3.28	3.30	3.35	3.35	3.38	3.27
2022	3.27	3.30	3.33	3.26	3.25	3.43	3.27	3.37	3.32	3.36	3.31	3.25

Figure 5: Monthly SR 520 Toll Fee from 2012-2022

d. Travel Time

We will only be using year 2021's data due to time limitations, the data contains 97 rows x 6 columns:

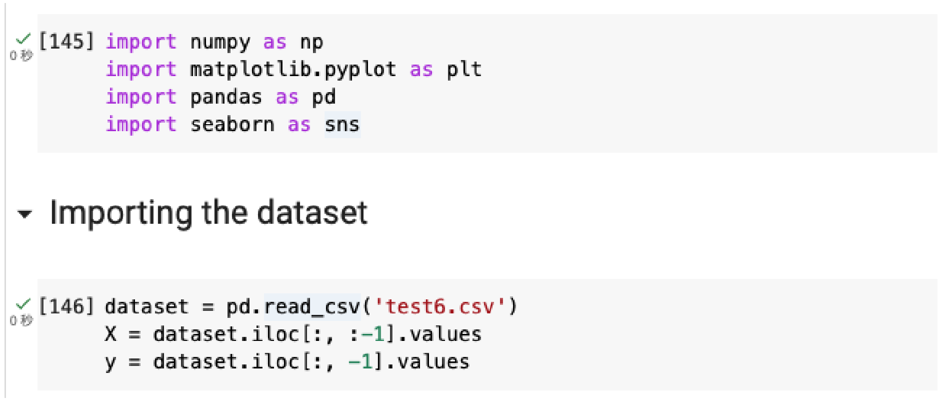

commute-time.xlsx

	A	B	C	D	E	F
1	YYYY-MM	Route	Direction	Time	Average travel time at peak (min)	95th percentile travel time(min)
2	2021-01	I-90	Seattle to Bellevue	Morning	9	15
3	2021-02	I-90	Seattle to Bellevue	Morning	10	11
4	2021-03	I-90	Seattle to Bellevue	Morning	10	11
5	2021-04	I-90	Seattle to Bellevue	Morning	10	12
6	2021-05	I-90	Seattle to Bellevue	Morning	10	12
7	2021-06	I-90	Seattle to Bellevue	Morning	11	12

Figure 6: I-90 Average Travel Time in 2021

2.4. Data Preprocessing

We started by Including all the above variables in a single dataset to feed our model. After importing all libraries as needed, we separate our dataset in X and y. X is our input of features, and y is our output (refer to Appendix 4.2 - 3).

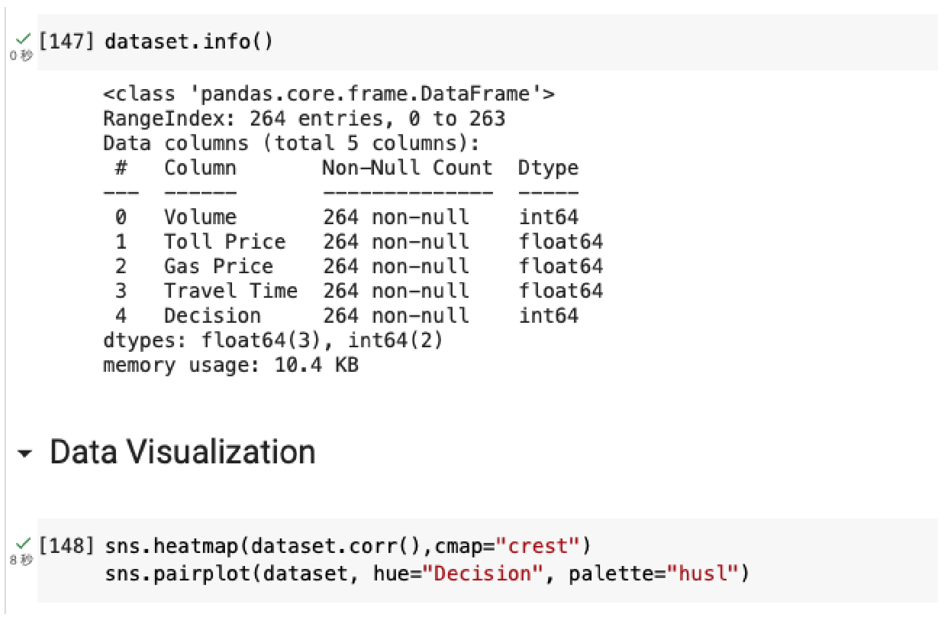

```
[145] import numpy as np
      import matplotlib.pyplot as plt
      import pandas as pd
      import seaborn as sns
```

Importing the dataset

```
[146] dataset = pd.read_csv('test6.csv')
      X = dataset.iloc[:, :-1].values
      y = dataset.iloc[:, -1].values
```

Figure 7: Importing dataset

To verify the validity of our dataset, we conducted measures of correlation. The results are as follow:



```
[147] dataset.info()
```

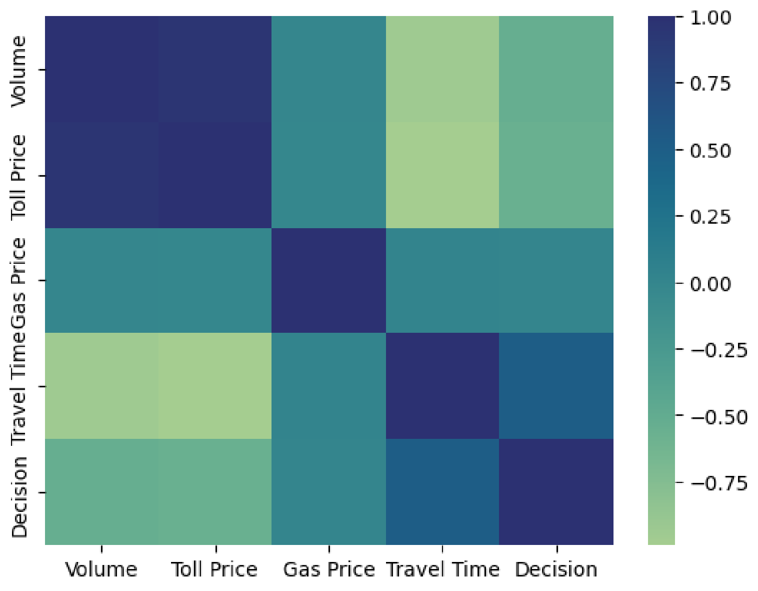
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264 entries, 0 to 263
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Volume       264 non-null    int64  
 1   Toll Price   264 non-null    float64
 2   Gas Price    264 non-null    float64
 3   Travel Time  264 non-null    float64
 4   Decision     264 non-null    int64  
dtypes: float64(3), int64(2)
memory usage: 10.4 KB
```

Data Visualization

```
[148] sns.heatmap(dataset.corr(),cmap="crest")
      sns.pairplot(dataset, hue="Decision", palette="husl")
```

Figure 8: dataset correlation

The below graphs show the correlation between every feature.



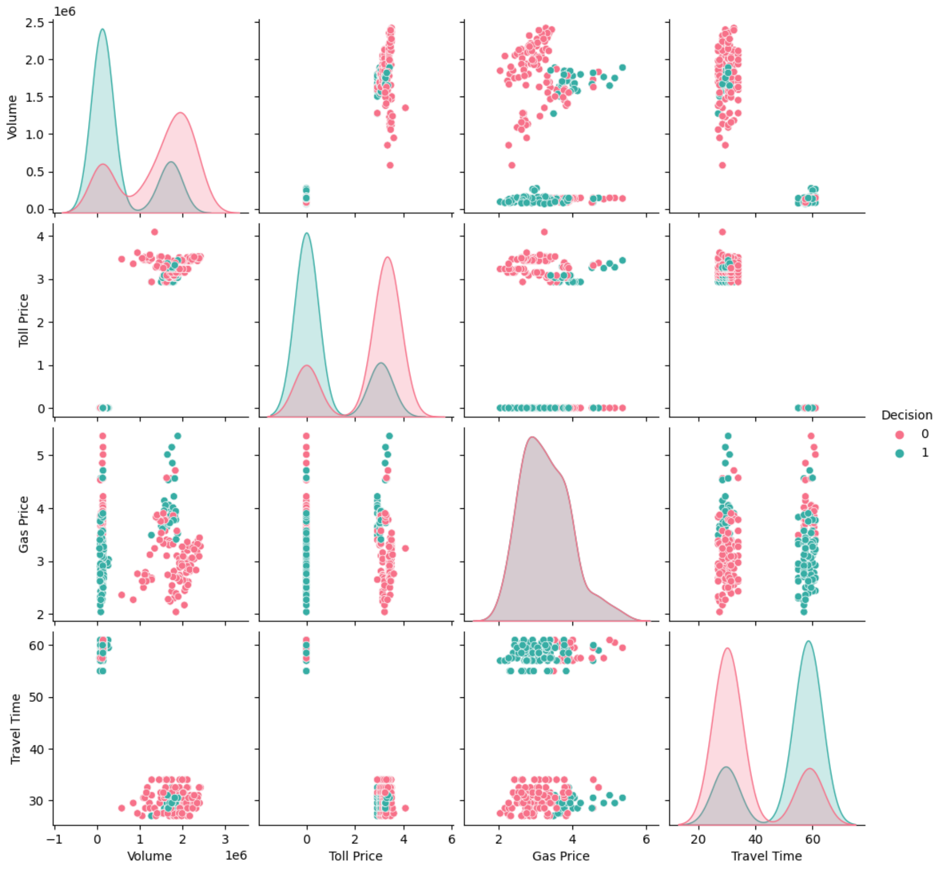


Figure 9: correlation between each variables

As showed, our dataset has several limitations,

1. The dataset includes both discrete and continuous data
2. The dataset presents non-normal distribution of features
3. Insufficient dataset
4. Lack of an objective way of presenting the output

Due to different scales of measure while collecting our data, some variables are presenting discrepancies. However, we will assume that they are normally distributed as how they behave in the real world. As for our expected output, we defined a formula to calculate the opportunity cost, the bridge with the lowest cost would be considered as the ideal one. The formula calculates the cost by summing up three terms, each weighted by a specific factor based on personal preferences:

*Cost ($) = W1 * Toll Price ($) + W2 * [Gas Price ($/gal) * Travel Distance (mil) * 1/Fuel Efficiency (mil/gal)] + W3 * [Travel Time (min) * Average Income ($/min)]*

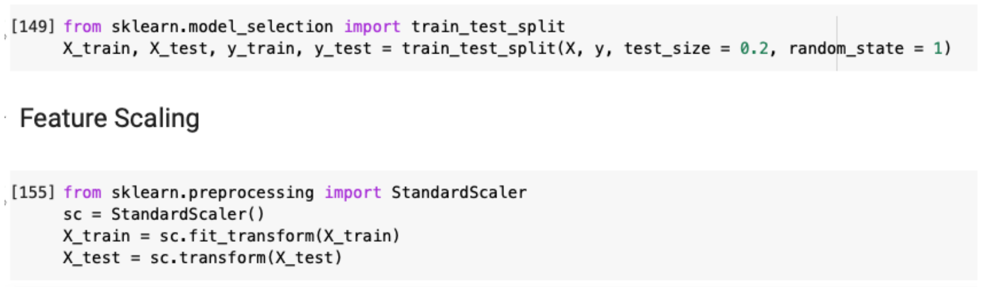
I90 travel_distance = 12 miles; 520 travel_distance = 8.7 miles

Const_fuel_efficiency = 25.7 mil/gal

Const_avg_income = 0.5 $/min

W1 : W2 : W3 = 1 : 1 : 0.15

Afterwards, we separate the data into training sets and test sets. Then apply feature scaling to put all features in the same scale, avoiding dominated features.



```
[149] from sklearn.model_selection import train_test_split
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 1)
```

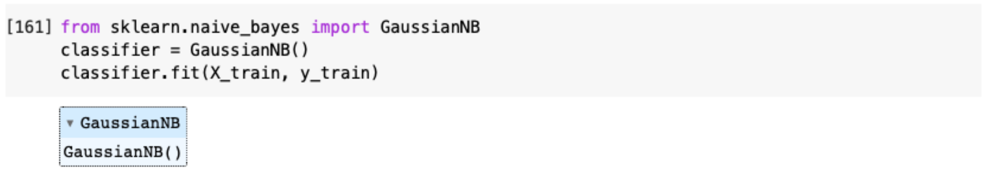
Feature Scaling

```
[155] from sklearn.preprocessing import StandardScaler
      sc = StandardScaler()
      X_train = sc.fit_transform(X_train)
      X_test = sc.transform(X_test)
```

Figure 10: Feature scaling

2.5. Method - Naive Bayes Classifier

By using Sklearn, we can simplify our calculations and reduce the time and effort required to build and test our models. Specifically, we create an instance of the GaussianNB model which applies concepts of Bayes' Theorem to compute the posterior probability of each class given the provided features. We then train the model on the training set and apply it to make predictions on the testing data. (refer to Appendix 4.2 - 3).

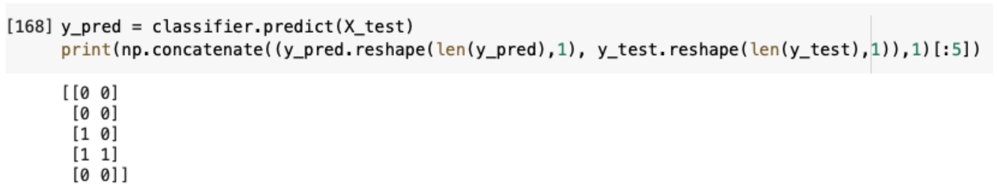


```
[161] from sklearn.naive_bayes import GaussianNB
      classifier = GaussianNB()
      classifier.fit(X_train, y_train)

      ▾ GaussianNB
      GaussianNB()
```

Figure 12: GaussianNB model

As a result, the class with the highest posterior probability is selected as the predicted class for the new data point.

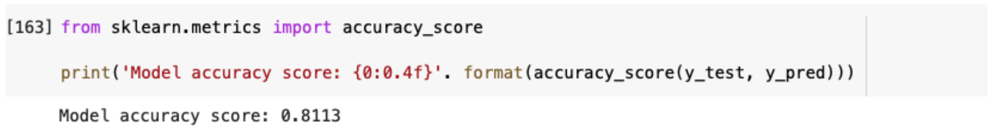


```
[168] y_pred = classifier.predict(X_test)
      print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1)[:5])

      [[0 0]
       [0 0]
       [1 0]
       [1 1]
       [0 0]]
```

Figure 13: Applying the model

Comparing the training set and what the model predicted for the test set, we can evaluate the accuracy of the model.



```
[163] from sklearn.metrics import accuracy_score

      print('Model accuracy score: {0:0.4f}'. format(accuracy_score(y_test, y_pred)))

      Model accuracy score: 0.8113
```

Figure 14: Evaluating the accuracy

The accuracy rate of our Naive Bayes Classifier is 0.81, indicating that the model correctly classifies which bridge we chose in 81 out of every 100 instances in the test data. In our case, it is not performing as well since the assumptions of Gaussian distribution are not met. The performance will enhance as we feed more reasonable data and tune variables.

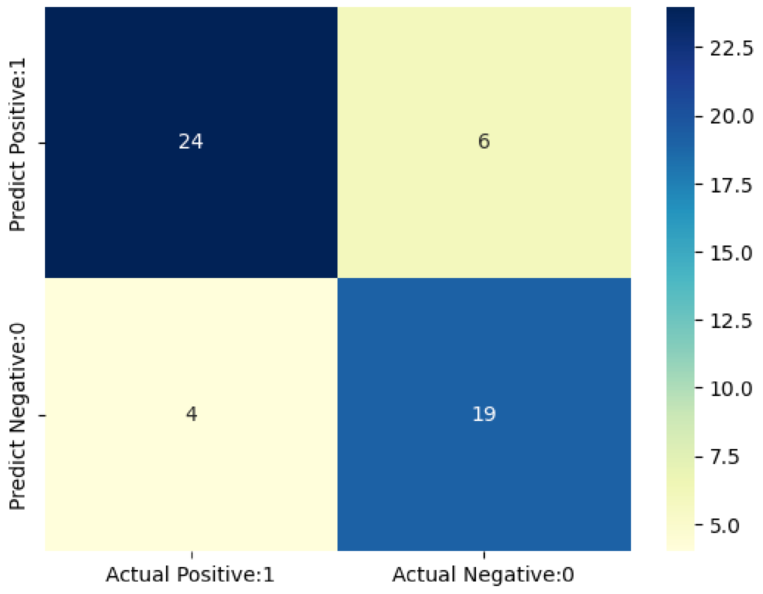
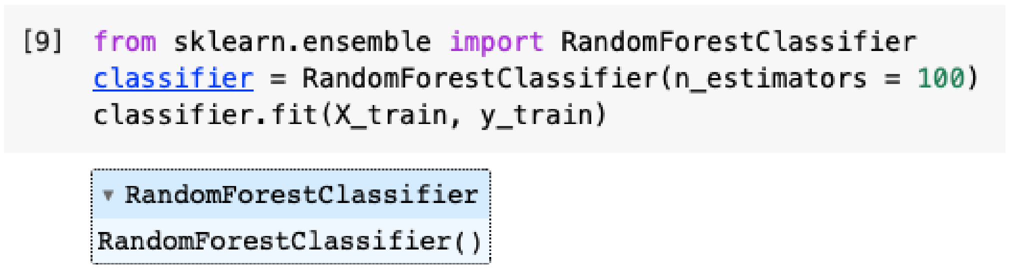


Figure 15: Confusion Matrix for Naive Bayes Classifier

2.6. Method - Random Forest Classifier

Another popular classification algorithm that we will use is the Random Forest Classifier. A key benefit of the model is its ability to handle complex relationships between the input features and the target variable.e (refer to Appendix 4.2 - 3).

To start, we create an instance of the RandomForestClassifier model and specify the number of decision trees to use.

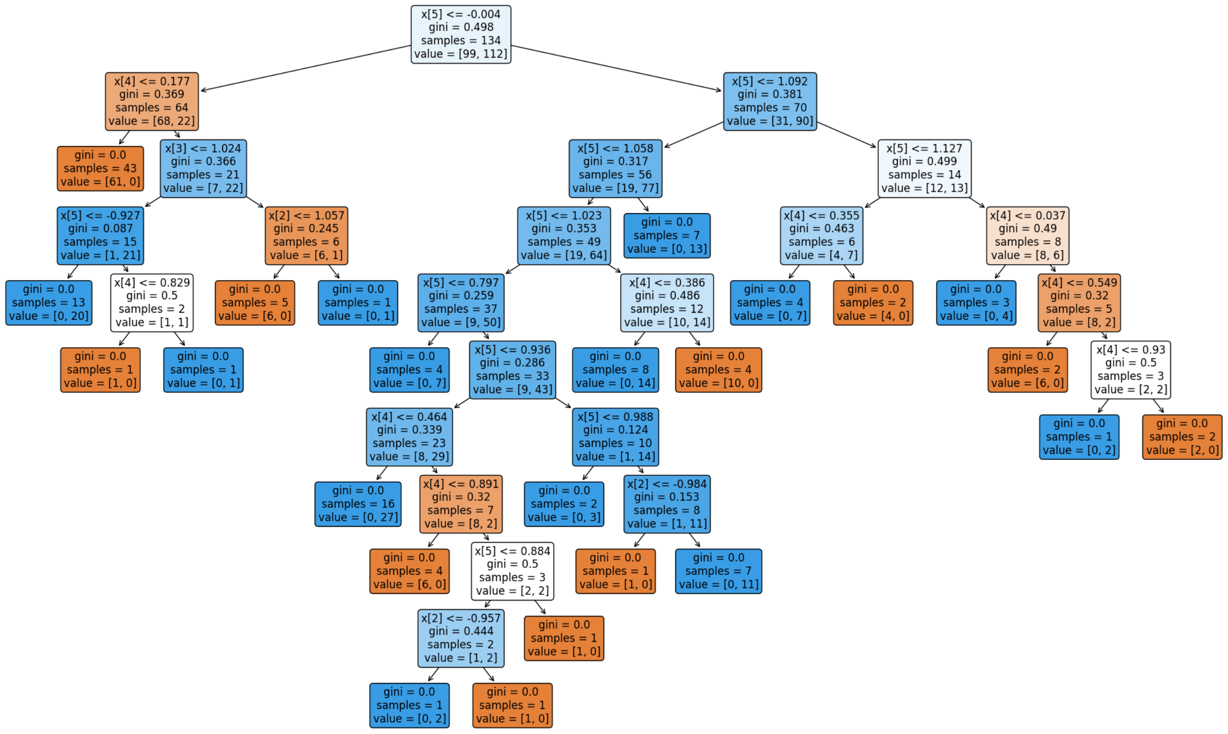


```
[9] from sklearn.ensemble import RandomForestClassifier
    classifier = RandomForestClassifier(n_estimators = 100)
    classifier.fit(X_train, y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

Figure 16: creating Random Forest Classifier

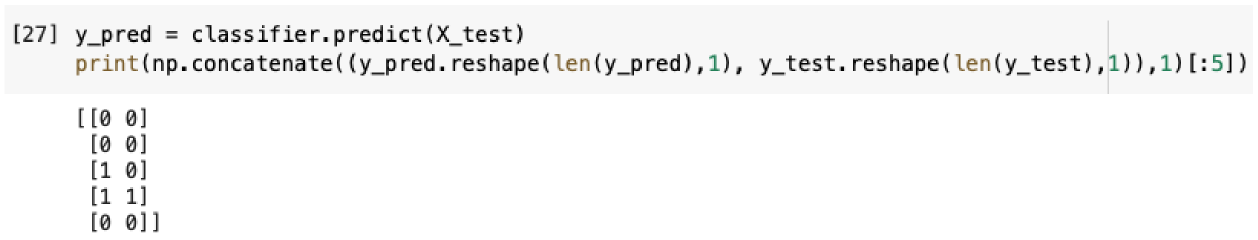
Since plotting all of our trees would require time and dedication, we simply plot just the five of them to have a look at how it differs from Naives Bayes Classifier.



x[5] <= -0.004 gini = 0.498 samples = 134 value = [99, 112]
x[4] <= 0.177 gini = 0.369 samples = 64 value = [68, 22]
x[5] <= 1.092 gini = 0.381 samples = 70 value = [31, 90]
gini = 0.0 samples = 43 value = [61, 0]
x[3] <= 1.024 gini = 0.366 samples = 21 value = [7, 22]
x[5] <= 1.058 gini = 0.317 samples = 56 value = [19, 77]
x[5] <= 1.127 gini = 0.499 samples = 14 value = [12, 13]
x[5] <= -0.927 gini = 0.087 samples = 15 value = [1, 21]
x[2] <= 1.057 gini = 0.245 samples = 6 value = [6, 1]
x[5] <= 1.023 gini = 0.353 samples = 49 value = [19, 64]
gini = 0.0 samples = 7 value = [0, 13]
x[4] <= 0.355 gini = 0.463 samples = 6 value = [4, 7]
x[4] <= 0.037 gini = 0.49 samples = 8 value = [8, 6]
gini = 0.0 samples = 13 value = [0, 20]
x[4] <= 0.829 gini = 0.5 samples = 2 value = [1, 1]
gini = 0.0 samples = 5 value = [6, 0]
gini = 0.0 samples = 1 value = [0, 1]
x[5] <= 0.797 gini = 0.259 samples = 37 value = [9, 50]
x[4] <= 0.386 gini = 0.486 samples = 12 value = [10, 14]
gini = 0.0 samples = 4 value = [0, 7]
gini = 0.0 samples = 2 value = [4, 0]
gini = 0.0 samples = 3 value = [0, 4]
x[4] <= 0.549 gini = 0.32 samples = 5 value = [8, 2]
gini = 0.0 samples = 1 value = [1, 0]
gini = 0.0 samples = 1 value = [0, 1]
gini = 0.0 samples = 4 value = [0, 7]
x[5] <= 0.936 gini = 0.286 samples = 33 value = [9, 43]
gini = 0.0 samples = 8 value = [0, 14]
gini = 0.0 samples = 4 value = [10, 0]
gini = 0.0 samples = 2 value = [6, 0]
x[4] <= 0.93 gini = 0.5 samples = 3 value = [2, 2]
x[4] <= 0.464 gini = 0.339 samples = 23 value = [8, 29]
x[5] <= 0.988 gini = 0.124 samples = 10 value = [1, 14]
gini = 0.0 samples = 1 value = [0, 2]
gini = 0.0 samples = 2 value = [2, 0]
gini = 0.0 samples = 16 value = [0, 27]
x[4] <= 0.891 gini = 0.32 samples = 7 value = [8, 2]
gini = 0.0 samples = 2 value = [0, 3]
x[2] <= -0.984 gini = 0.153 samples = 8 value = [1, 11]
gini = 0.0 samples = 4 value = [6, 0]
x[5] <= 0.884 gini = 0.5 samples = 3 value = [2, 2]
gini = 0.0 samples = 1 value = [1, 0]
gini = 0.0 samples = 7 value = [0, 11]
x[2] <= -0.957 gini = 0.444 samples = 2 value = [1, 2]
gini = 0.0 samples = 1 value = [1, 0]
gini = 0.0 samples = 1 value = [0, 2]
gini = 0.0 samples = 1 value = [1, 0]

Figure 17: plotting all of out trees

We then train the model on the training set



```
[27] y_pred = classifier.predict(X_test)
     print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1)[:5])

     [[0 0]
      [0 0]
      [1 0]
      [1 1]
      [0 0]]
```

Figure 18: training model on the training set

After training, we use the Random Forest Classifier to make predictions on the testing data.

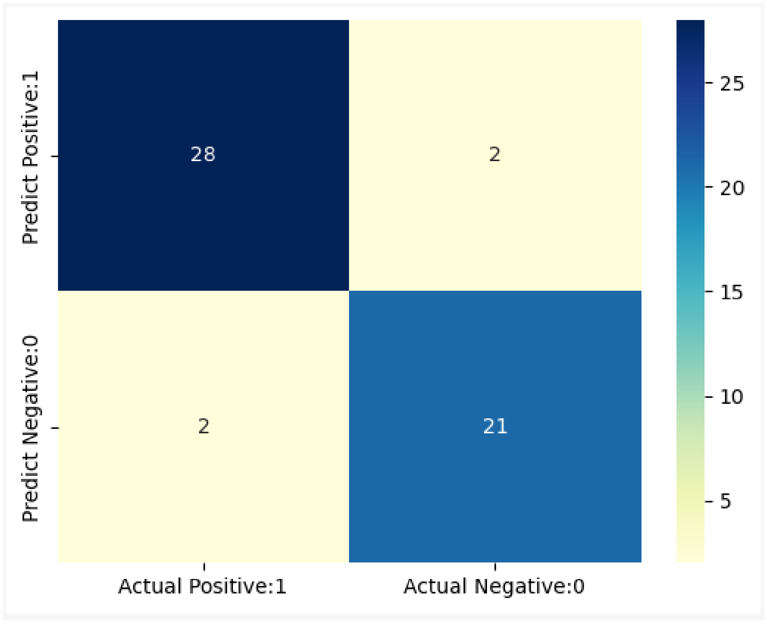
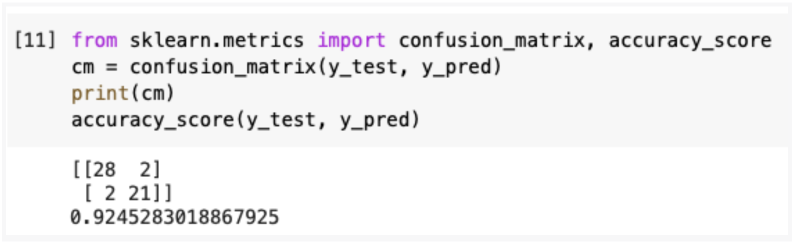


Figure 19: Confusion Matrix for Random Forest Classifier



```
[11] from sklearn.metrics import confusion_matrix, accuracy_score
     cm = confusion_matrix(y_test, y_pred)
     print(cm)
     accuracy_score(y_test, y_pred)

     [[28  2]
      [ 2 21]]
     0.9245283018867925
```

Figure 20: Evaluating the accuracy

The accuracy rate is 0.92, which is significantly better than our previous model's accuracy of 0.81. Moreover, we evaluate the relative importance of each feature in the Random Forest Classifier to identify the most significant factors in making the decision. By looking at the feature importance plot, we observe that the price, combining our toll price and gas price together, is the most important feature, followed by volume and travel time.

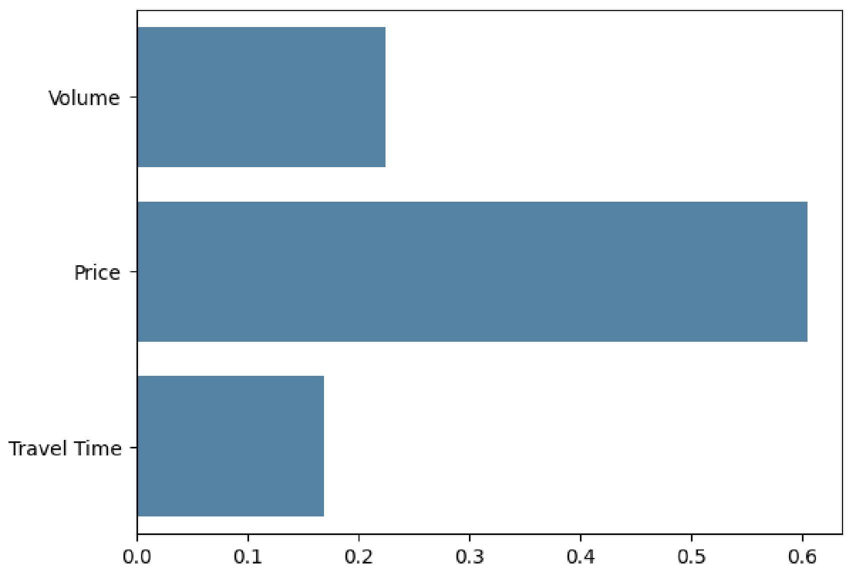


Figure 21: Feature importance plot

2.7. Model Results

In conclusion, after comparing the performance between the models, we found that the Random Forest Classifier is more effective at classifying which bridge to choose; it provides a higher level of precision in predicting the outcomes.

3. Conclusion

3.1. Answer question

Base on the model result, we can answer our question:

1. We will use Random Forest Classifier to decide if which bridge is a better choice: SR-520 or I-90” given a random datapoint with traffic volume, toll price, gas price and travel time
2. Price is the most important variable among all, followed by travel time and volume

3.2. Weakness and limitation

One of the limitations of the project is the source and size of data. When collecting traffic volume data for 520 bridge, we were only able to find the average monthly translation from 2012 to 2022 instead of daily or hourly data which may provide more information for results that are closer to reality. Due to the limitation of traffic volume data, variables’ data was averaged to monthly bases for calculations. This might also affect the accuracy of the models and the results could be too vague since in real life, traffic volume, gas price, traffic time, and toll fee are changing constantly, daily, hourly, or even minutely.

To fix the weakness for future research, a larger dataset with more specific time intervals could possibly improve the results from the model. To further improve the analysis, factors other than traffic volume, gas, and toll fee rates like weather, accidents, special events, gas price, and even inflation rates could be taken in as other variables.

Another limitation is the short timeframe. Given the short timeframe, we did not examine the impact of factors other than traffic volume, gas, toll fee rates, and travel time like weather conditions or accidents on travel time, as this would require additional data sources and more extensive analysis, which would have impact on the accuracy of the models.

Lastly, optimizing both models by finding the values of the hyperparameters could result in better performance. In the meantime, there are more machine learning models or more advanced modeling techniques to improve the accuracy of predictions.

However, once generated fine models are fed with more granular data, it should be able to be further revised and improved to take in more variables, like weather, accidents, special events, gas price, and even inflation rates.

3.3. What we learned from the project

Peihsuan Lin:

I learned how Bayes Theorem solves problems. While it is typically introduced in class using examples with one or two variables, there are often several factors that can affect the outcome under real-world scenarios. In addition, I learn how machine learning algorithms figure patterns given a dataset, such as splitting the data into different sets. Furthermore, although python libraries simplified the meticulous equations, our data validity also matters. We’ve made assumptions about our datasets, but it is often challenging to obtain all as anticipated. Lastly, even though we didn't fine-tune the parameters and approach the problem in a general way, it was still amazing to see the power of machine learning in action compared to traditional statistical methods. This course work gave me a chance to learn the basics of classification models and I feel inspired to delve into others.

Huiru Yang:

This project has taught me a lot about the fundamental role of Bayes' theorem in building complex models. Through the analysis of traffic flow data and relevant variables, we found that the intuition about the most important factor may not always be correct. The results of the Bayes model and other models such as the random forest showed that toll price and gas price played a more significant role in choosing the optimal route, followed by travel time and volume. This was contrary to my initial assumption that travel time data would be the most important factor. Overall, this project has provided valuable insights into the benefits of using probability theory and Bayesian models in solving real-world problems.

Yi-Chia Chu:

Through this project, I have learned the skill of identifying everyday problems and utilizing classroom knowledge to design effective solutions. Our team utilized Bayes' theorem to analyze data and applied the Naive Bayesian model. And with the random forest model, we were able to identify the key variables for analysis. The process of data collection and interpretation taught us that unexpected results are not uncommon, and we had to adapt our solution design accordingly. We also learned to use the models' results to explain any contrary to our original assumptions. Overall, this project was a valuable opportunity to apply formula learned in class to real-world problem-solving and analysis.

4. Appendix

4.1. Reference

1) Monthly Traffic Volume:

SR520: https://wsdot.wa.gov/about/accountability/tolling-reports-policy

I90: https://wsdot.public.ms2soft.com/tcds/tsearch.asp?loc=Wsdot&mod=TCDS

2) Monthly Regular Gas Price: https://data.rgj.com/gas-price/washington/SWA/2022-09-26/

3) Toll Fee for SR-520: https://wsdot.wa.gov/about/accountability/tolling-reports-policy

4) Travel Time:

SR520:

https://wsdot.wa.gov/about/data/multimodal-mobility-dashboard/dashboard/central-puget-sound/stateroute520-cps/commute-time.htm

I90:
https://wsdot.wa.gov/about/data/multimodal-mobility-dashboard/dashboard/central-puget-sound/interstate90-cps/commute-time.htm

4.2. Source

1) .py file for retrieving data from .xlsx file (steps specified in the comments)

```python
import pandas as pd
def get_data(file_name: str, sheet_name) -> pd.DataFrame:
    # read the excel file, and the first sheet
    df = pd.read_excel(file_name, sheet_name)
    # get the disired data
    df = df.iloc[9:, 25].values
    # convert empty values
    df = [0 if pd.isnull(x) else x for x in df]
    return df
file_names = ['MonthlyVolumeReport_R017_1_2012.xlsx',
              'MonthlyVolumeReport_R017_1_2013.xlsx',
              'MonthlyVolumeReport_R017_1_2014.xlsx',
              'MonthlyVolumeReport_R017_1_2015.xlsx',
              'MonthlyVolumeReport_R017_1_2016.xlsx',
              'MonthlyVolumeReport_R017_4_2017.xlsx',
              'MonthlyVolumeReport_R017_1_2018.xlsx',
              'MonthlyVolumeReport_R017_1_2019.xlsx',
              'MonthlyVolumeReport_R017_1_2020.xlsx',
              'MonthlyVolumeReport_R017_1_2021.xlsx',
              'MonthlyVolumeReport_R017_7_2022.xlsx',]
# initialize an empty list to store the data
monthly_volume = []
# loop through the file names and sheet names
for file_name in file_names:
    # get the hourly data for the given file and sheet
    df = get_data(file_name, 0)
    # append the data to the list
    monthly_volume.append(df)
# Create a DataFrame store month traffic volume data from 2012 to 2022
data = {'Year': [2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022],
    'January': [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100],
    'February': [200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200],
    'March': [300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300],
    'April': [400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400],
    'May': [500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500],
    'June': [600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500, 1600],
    'July': [700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500, 1600, 1700],
    'August': [800, 900, 1000, 1100, 1200, 1300, 1400, 1500, 1600, 1700, 1800],
    'September': [900, 1000, 1100, 1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900],
    'October': [1000, 1100, 1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900, 2000],
    'November': [1100, 1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900, 2000, 2100],
    'December': [1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900, 2000, 2100, 2200]}
# update the data with the actual data stored in the list monthly_volume
for i in range(len(data['Year'])):
    for j in range(1, len(data.keys())):
        data[list(data.keys())[j]][i] = monthly_volume[i][j-1]
# create a DataFrame
monthly_traffic = pd.DataFrame(data)
monthly_traffic.set_index('Year', inplace=True)
# write the DataFrame to a csv file
monthly_traffic.to_csv('I90_monthly_traffic.csv')
```

2) .py file for retrieving data from website (steps specified in the comments)

```python
from requests_html import HTMLSession
from bs4 import BeautifulSoup
import pandas as pd

def get_table_from_web(url):
    # create an HTMLSession object
    session = HTMLSession()
    # get the website content
    r = session.get(url)
    # render the page
    r.html.render()
    # use BeautifulSoup to find tables on the page
    soup = BeautifulSoup(r.html.html, 'html.parser')
    return soup.find_all('table')

def convert_table_to_csv(table, filename):
    # convert the table to a pandas dataframe
    df = pd.read_html(str(table))[0]
    # write into a csv file
    df.to_csv(filename, index=False)
```

```
# washington gas price data
url = 'https://data.rgj.com/gas-price/washington/SWA/2022-09-26/'
washington_all_tables = get_table_from_web(url)
washington_table = washington_all_tables[1]
convert_table_to_csv(washington_table, 'washington-regular-gas-data.csv')
```

3) .py file for Naive Bayes & Random Forest Classifier (steps specified in the comments)

```python
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
# Importing the dataset
dataset = pd.read_csv('raw_data.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
dataset.info()
# Data Visualization
sns.heatmap(dataset.corr(),cmap="crest")
sns.pairplot(dataset, hue="Decision", palette="husl")
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state =1)
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
# Naive Bayes
# Training the Naive Bayes model on the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
# Predicting the Test set results
```

```
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),
                      y_test.reshape(len(y_test),1)),1)[:5])

# Check accuracy score
from sklearn.metrics import accuracy_score
print('Model accuracy score: {0:0.4f}'. format(accuracy_score(y_test, y_pred)))
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
# Visualizing the Confusion Matrix
cm_matrix = pd.DataFrame(data=cm, columns=['Actual Positive:1', 'Actual Negative:0'],
                                 index=['Predict Positive:1', 'Predict Negative:0'])
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')
# Random Forest
# Training the Random Forest Classification model on the Training set
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 100)
classifier.fit(X_train, y_train)
# Pull out one tree from the forest
import pydot
Tree = classifier.estimators_[5]
# Export the image to a dot file
from sklearn import tree
plt.figure(figsize=(25,15))
tree.plot_tree(Tree,filled=True,
              rounded=True,
              fontsize=12);
# Predicting the Test set results
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1)[:5])
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
# Visualizing the Confusion Matrix
import seaborn as sns
```

```
cm_matrix = pd.DataFrame(data=cm, columns=['Actual Positive:1', 'Actual Negative:0'],
                                 index=['Predict Positive:1', 'Predict Negative:0'])
sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')
# Making a feature importance plot
feature_importances = classifier.feature_importances_
feature_names = ["Volume", "Price", "Travel Time"]
sns.barplot(x=feature_importances,y=feature_names, color='steelblue')
```