Which bridge is a better choice:
SR-520 or I-90?

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

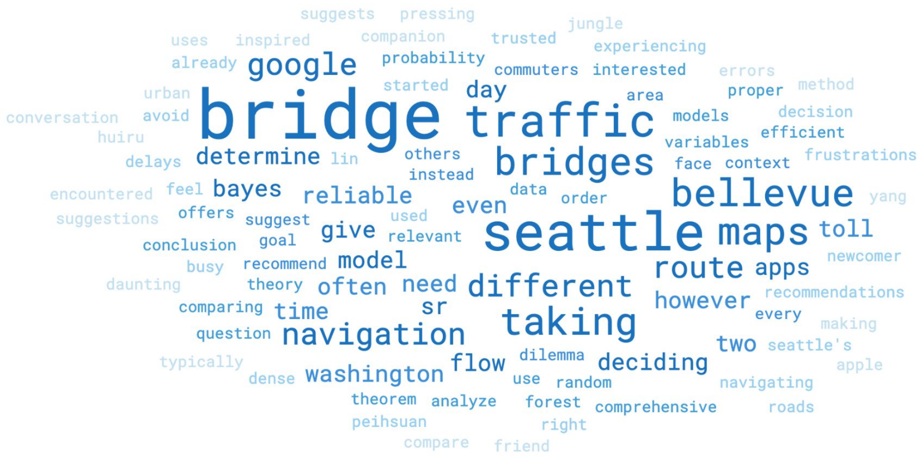
CS 5001 Discrete Structures Final Project

Introduction

Our Question:

“Which bridge is a better choice? SR-520 or I-90?”

➔ How the question relate to three of us



Introduction

Relation to the course:

- <u>**Conditional Probability**</u>
- Bayes' theorem

P(A given B) or P(A | B)

P(I90 is selected | gas price is at x number)

P(SR520 is selected | gas price is at x number)

P(I90 is selected | there has construction going on)

P(SR520 is selected | there has construction going on)

P1 (SR520 is selected | variable x)

P1 (I90 is selected | variable x)

P2 (SR520 is selected | variable y)

P2 (I90 is selected | variable y)

And so on……

Introduction

Relation to the course:

- Conditional Probability
- **Bayes' theorem**

P(A|B) =

P(B|A)*P(A)/(P(B|A)*P(A) + P(B|not A)*P(not A))

OR P(A|B) =

P(B | A) * P(A) / P(B)

P(SR520 is selected|variable x) =

P(variable x | SR520 is selected) * P(SR520 is selected) / P(variable x)

P(I90 is selected|variable x) =

P(variable x | I90 is selected) * P(I90 is selected) / P(variable x)

Analysis

Binary Classification:

- the direct application of Bayes Theorem will become intractable, especially when the number of variables increases.

Take ‘SR-520 is selected’ as an example:

P(SR520 is selected|variable x) =

P(variable x | SR520 is selected) * P(SR520 is selected) / P(variable x)

P(SR520 is selected|variable y) =

P(variable y | SR520 is selected) * P(SR520 is selected) / P(variable y)

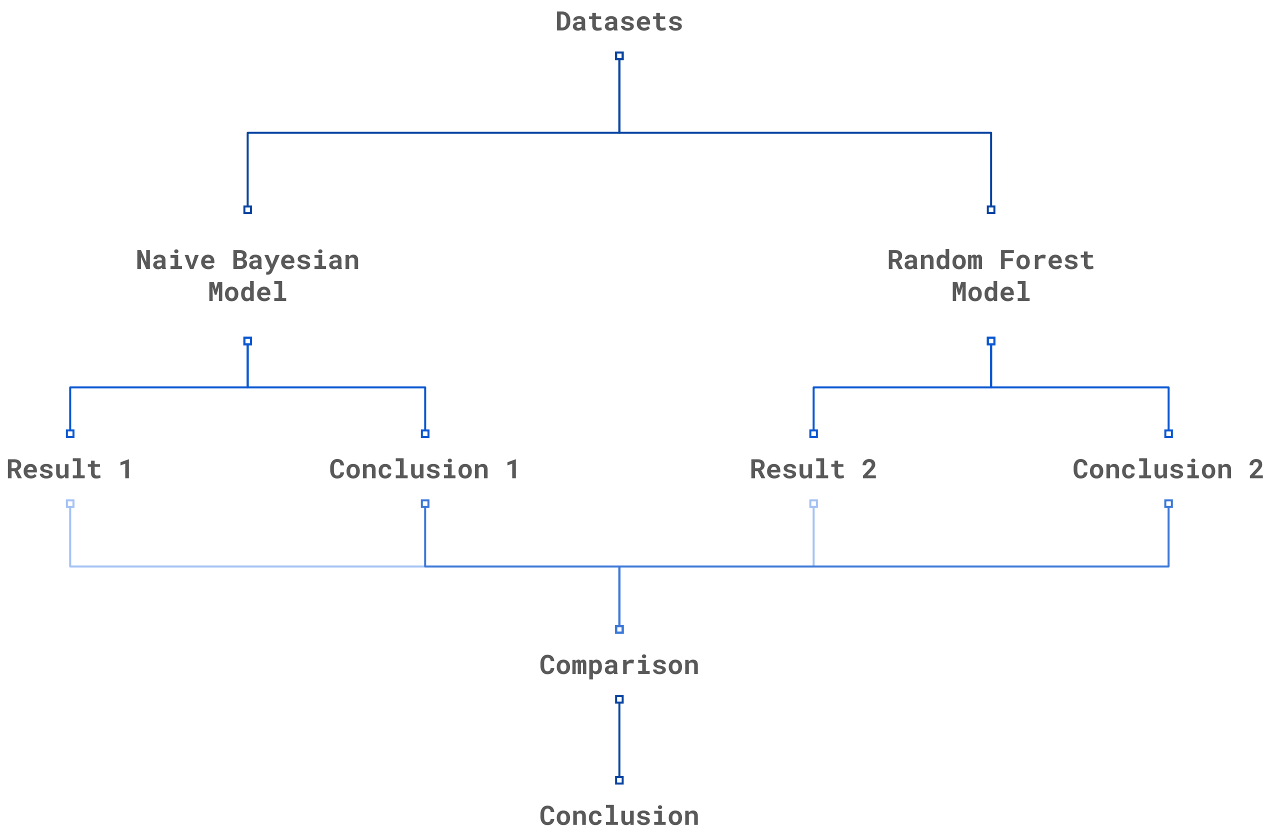
P(SR520 is selected|variable z) =

P(variable z | SR520 is selected) * P(SR520 is selected) / P(variable z)

....

Analysis

Our Methodology



Analysis

Dataset

- **Monthly Traffic Volume**
- Monthly Regular Gas Price
- Toll Fee
- Travel Time

I-90 Monthly Traffic Volume

- Source: Traffic Count Database System (TCDS) operated by WSDOT
- Time Period: 2012 to 2022
- Total Count(monthly average hourly traffic volume)

In [ ]:

```
import pandas as pd

def get_data(file_name: str, sheet_name) -> pd.DataFrame:
    # with directory specified
    directory = '/Users/huiruyang/Documents/Huiru_info/00_NEU/2023 1st Semester/CS5002/FinalProject/Traf
    # read the excel file, and the first sheet
    df = pd.read_excel(directory + file_name, sheet_name)
    # get the values which is saved in the column Z (total) and start from row 11 from the excel file
    df = df.iloc[9:, 25].values
    # convert empty values to 0
    df = [0 if pd.isnull(x) else x for x in df]
    return df
```

Out[ ]:

Year	January	February	March	April	May	June	July	August	September	October	November	December
2012	84390.0	87903.0	136382.0	136566.0	145242.0	144524.0	110876.0	95218.0	135232.0	141788.0	143838.0	146001.0
2013	78623.0	131184.0	138648.0	139837.0	100242.0	84497.0	131661.0	134447.0	136682.0	140561.0	141441.0	108102.0
2014	74965.0	125549.0	133818.0	102337.0	85467.0	132259.0	134760.0	136591.0	140335.0	143227.0	97010.0	86033.0
2015	76989.0	113819.0	96195.0	78631.0	129510.0	138417.0	142294.0	144345.0	147297.0	106421.0	87701.0	137697.0
2016	81660.0	91791.0	78338.0	127220.0	132107.0	140348.0	142503.0	143124.0	NaN	NaN	NaN	135569.0
2017	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	141594.0
2018	75071.0	129407.0	136637.0	136494.0	NaN	99812.0	82361.0	133864.0	133660.0	139174.0	133186.0	NaN
2019	87337.0	141565.0	145893.0	148452.0	114651.0	NaN	148992.0	153956.0	155886.0	160711.0	161328.0	123008.0
2020	85549.0	138810.0	147012.0	112294.0	92386.0	148894.0	156885.0	161890.0	161634.0	159980.0	118635.0	91992.0
2021	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	109452.0	80386.0	NaN	NaN
2022	140500.0	105464.0	86100.0	83067.0	127647.0	136782.0	144167.0	144761.0	NaN	NaN	134166.0	142877.0

Analysis

Dataset

- **<u>Monthly Traffic Volume</u>**
- Monthly Regular Gas Price
- Toll Fee
- Travel Time

SR-520

Source: Tolling reports & policy: Toll Traffic and Revenue from Washington State Department of Transportation (WSDOT)

Time Period: 2012 to 2022

Monthly Traffic Volume: Reported transaction in each month

	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

Analysis

Dataset

- Monthly Traffic Volume
- **Monthly Regular Gas Price**
- Toll Fee
- Travel Time

YYYY-M	Reg. Gasoline Price
2022-12	3.97
2022-11	4.57
2022-10	5.01
2022-09	4.56
2022-08	4.71
...	...
2012-05	4.22
2012-04	4.14
2012-03	4.01
2012-02	3.66
2012-01	3.49

132 rows × 1 columns

Monthly Reg.gas Price (WA)

- Source: database operated by Reno Gazette Journal
- Time Period: 2012 to 2022
- Average(Total Count(weekly average reg gasoline price))
- 1022 rows × 2 columns

```python
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')
```

Analysis

Dataset

- Monthly Traffic Volume
- Monthly Regular Gas Price
- **Toll Fee**
- Travel Time

SR-520

Source: Tolling reports & policy: Toll Traffic and Revenue from Washington State Department of Transportation (WSDOT)

Time Period: 2012 to 2022

Monthly Traffic Volume:

Reported values / Reported Transactions

	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

Analysis

Dataset

- Monthly Traffic Volume
- Monthly Regular Gas Price
- Toll Fee
- **Travel Time**

Morning and evening peak commute time

Source: Multimodal mobility dashboard's Commute times from WSDOT

Time Period: 2021,

Region: between Seattle and Bellevue

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

Analysis

Our formula

- Monthly Traffic Volume
- Monthly Regular Gas Price
- Toll Fee
- Travel Time

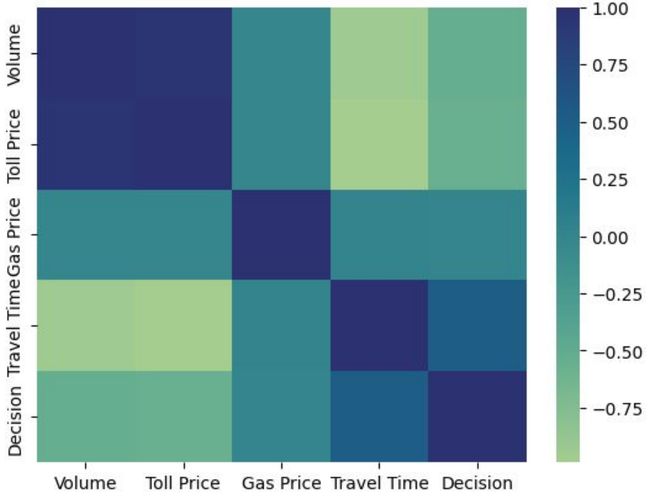
Cost ($) =

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

- I-90 travel_distance = 12 miles;
- SR-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

Analysis

- Dataset Overview



Feature correlation

```
[144] 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
```

Analysis

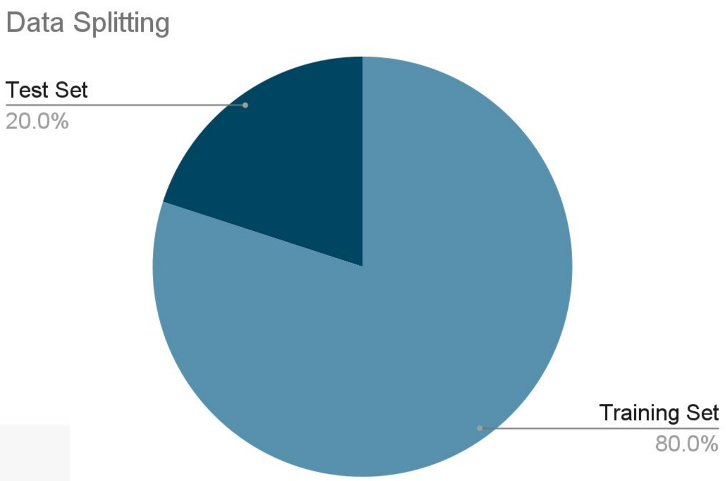
- Data Preprocessing

▾ Splitting the dataset into the Training set and Test set

```
[25] 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

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



Models - **Naive Bayes Classifier**

1. Training the model on the training set

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

```
▾ GaussianNB
GaussianNB()
```

2. Predicting test sets result

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

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

What the model predicts

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

Expected result - training set

0 is for choosing SR520, 1 is for choosing i90

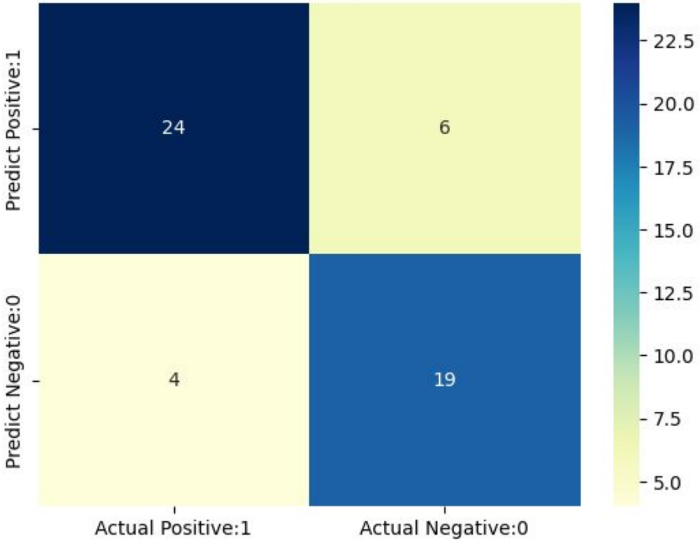
3. Check accuracy score

```
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
```

Accuracy score = 0.81



Confusion Matrix for Naive Bayes Classifier

Models - <u>**Random Forest Classifier**</u>

1. Training the model on the training set

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

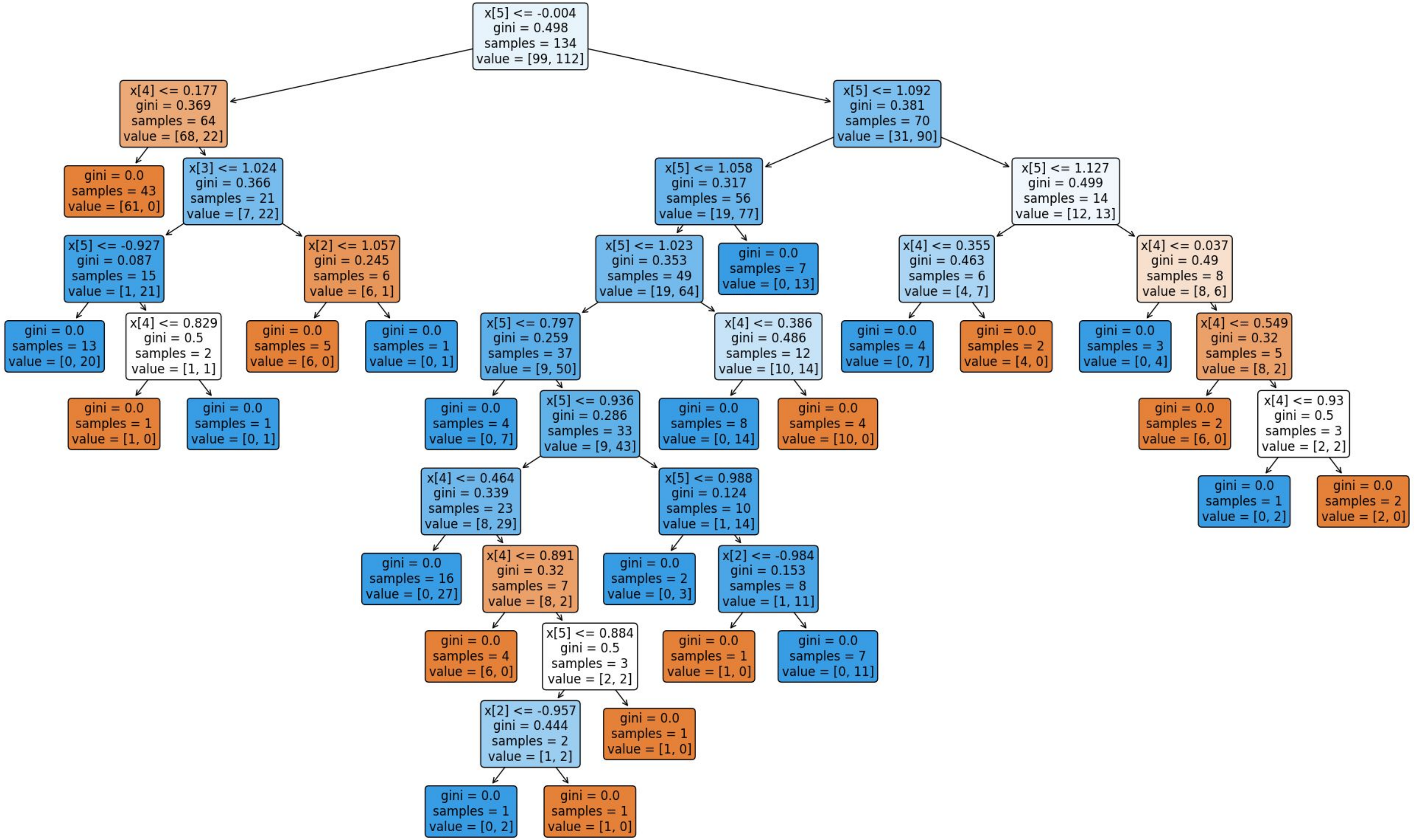
```
▾ RandomForestClassifier
RandomForestClassifier()
```

2. Predicting test sets result

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

3. Pull one tree to visualize

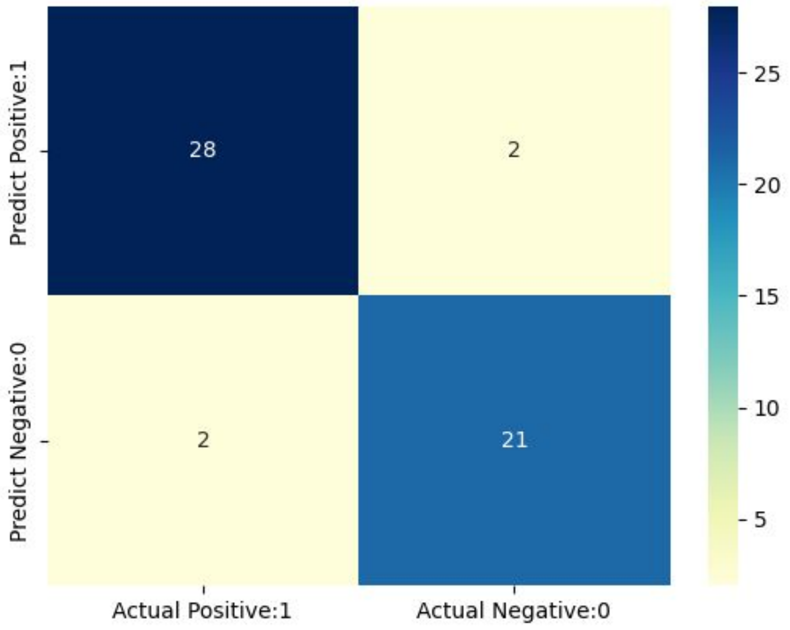


4. Check accuracy score

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
accuracy_score(y_test, y_pred)
```

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

Accuracy score = 0.92



Confusion Matrix for Random Forest Classifier

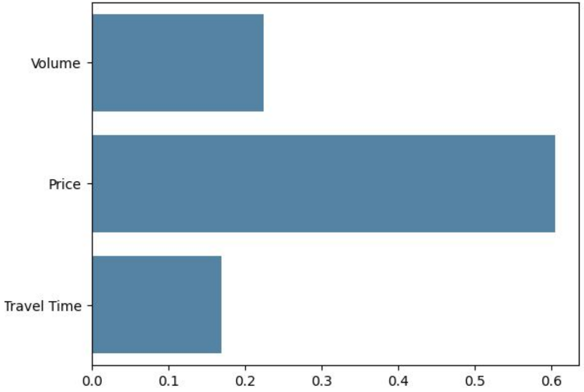
Results

Accuracy score

Naive Bayes 0.81 < 0.92 Random Forest

For the most important feature:

1. Price (Toll + Gas)
2. Traffic Volume
3. Travel Time



Conclusion

Answer to the question

➔ We now can answer the question of “Which bridge is a better choice: SR-520 or I-90” using the **Random Forest Classifier**

➔ **Price** is the most important feature, followed by travel time and traffic volume

Conclusion

Limitation

- the source and size of data
- the short timeframe
- accuracy of predictions

Action

➔ a larger dataset with more specific time intervals
➔ more extensive analysis
➔ finding the values of the hyperparameters

Conclusion

Takeaways