Logistic Regression on Uber and Lyft 2018 Dataset Boston, MA

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report, accurac
```

1. Load and Inspect the Datasets (Raw and Cleaned)

```
# Load raw data
df = pd.read_csv('/content/drive/MyDrive/DS160/Final Project/Myhre_Chris_Dat
df.head()
```

The raw dataset had 57 columns while the cleaned dataset has 28 columns. 29 columns were removed in excel due to their redundancy.

```
# Load cleaned data
df = pd.read_csv('/content/drive/MyDrive/DS160/Final Project/Myhre_Chris_Dat
df.head()
```

2. Data Overview and Missing Values

```
df.info()
df.describe()
df.isna().sum()
```

```
# Drop missing price columns (55095/693071 = 8% of data dropped)
df=df.dropna()
df.isna().sum()
```

3. Exploratory Data Analysis (EDA)

```
# Correlation Heatmap
plt.figure(figsize=(25,10))
sns.heatmap(df.select_dtypes(include=[np.number]).corr(), annot=True, cmap='
plt.title('Correlation Heatmap')
plt.show()

# Example distribution
sns.histplot(df['price'], kde=True, bins=30)
plt.title('Price Distribution')
plt.show()

plt.figure(figsize=(10, 6))
sns.countplot(x='weather', data=df)
plt.title('Distribution of Short Summary')
plt.show()
```

Heatmap Insights: Some notable correlations are dewPoint + humidity (0.75), temp + dewPoint (0.86), PrecipProbability + visibility (-0.76), but surprsingly no correlations between price and weather.

Histogram: Price seems to be concentrated around 10 dollars, few rides above 35 dollars.

Countplot: Most rides were ordered with overcast weather and second most rides ordered with mostly cloudy weather and third most rides ordered with partly cloudy weather. Contrary to what I thought, only a minority of rides were ordered during rainy weather.

```
df['weather'].value_counts()
```

Adding another price category to convert numerical values into categorical (High/Low).

```
median_price = df['price'].median()
df['price_cat'] = df['price'].apply(lambda x: 'High' if x > median_price els
df['price_cat'].value_counts()
```

Converting categorical columns into numerical

```
categorical_cols = df.select_dtypes(include=['object']).columns
df = pd.get_dummies(df, columns=categorical_cols, drop_first=True)
```

```
df.sample()
```

```
df.columns
```

```
Index(['hour', 'day', 'month', 'price', 'distance', 'surge_multiplier',
       'temperature', 'precipIntensity', 'precipProbability', 'humidity',
       'windSpeed', 'windGust', 'visibility', 'temperatureHigh',
       'temperatureLow', 'dewPoint', 'pressure', 'windBearing',
'cloudCover',
       'uvIndex', 'ozone', 'moonPhase', 'precipIntensityMax',
       'source_Beacon Hill', 'source_Boston University', 'source_Fenway',
       'source_Financial District', 'source_Haymarket Square',
       'source_North End', 'source_North Station',
       'source_Northeastern University', 'source_South Station',
       'source_Theatre District', 'source_West End', 'destination_Beacon
Hill',
       'destination_Boston University', 'destination_Fenway',
       'destination_Financial District', 'destination_Haymarket Square',
       'destination_North End', 'destination_North Station',
       'destination_Northeastern University', 'destination_South Station',
       'destination_Theatre District', 'destination_West End',
'cab_type_Uber',
       'name_Black SUV', 'name_Lux', 'name_Lux Black', 'name_Lux Black XL',
       'name_Lyft', 'name_Lyft XL', 'name_Shared', 'name_UberPool',
       'name_UberX', 'name_UberXL', 'name_WAV', 'weather_ Drizzle ',
       'weather_ Foggy ', 'weather_ Light Rain ', 'weather_ Mostly Cloudy ',
       'weather_ Overcast ', 'weather_ Partly Cloudy ',
       'weather_ Possible Drizzle ', 'weather_ Rain ', 'price_cat_Low'],
      dtype='object')
```

Model Elements Key

(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52,

53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66)

1. hour
2. day
3. month
4. price
5. distance
6. surge_multiplier
7. temperature
8. precipIntensity
9. precipProbability
10. humidity
11. windSpeed
12. windGust
13. visibility
14. temperatureHigh
15. temperatureLow
16. dewPoint
17. pressure
18. windBearing
19. cloudCover
20. uvIndex
21. ozone
22. moonPhase
23. precipIntensityMax
24. source_Beacon Hill
25. source_Boston University
26. source_Fenway
27. source_Financial District
28. source_Haymarket Square
29. source_North End
30. source_North Station
31. source_Northeastern University
32. source_South Station
33. source_Theatre District
34. source_West End
35. destination_Beacon Hill
36. destination_Boston University

37. destination_Fenway
38. destination_Financial District
39. destination_Haymarket Square
40. destination_North End
41. destination_North Station
42. destination_Northeastern University
43. destination_South Station
44. destination_Theatre District
45. destination_West End
46. cab_type_Uber
47. name_Black SUV
48. name_Lux
49. name_Lux Black
50. name_Lux Black XL
51. name_Lyft
52. name_Lyft XL
53. name_Shared
54. name_UberPool
55. name_UberX
56. name_UberXL
57. name_WAV
58. weather_ Drizzle
59. weather_ Foggy
60. weather_ Light Rain
61. weather_ Mostly Cloudy
62. weather_ Overcast
63. weather_ Partly Cloudy
64. weather_ Possible Drizzle
65. weather_ Rain

4. Handle Missing Values

For missing values I dropped the 8% of rows that had missing price values, there were no other missing values in the other columns.

5. Train-Test Split

```
df.sample()
```

```
X = df.drop(['price_cat_Low','price'], axis = 1)
y = df['price_cat_Low']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran
```

6. Train Logistic Regression Model

```
model = LogisticRegression(max_iter=10000)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

```
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_logistic.py:465
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
  n_iter_i = _check_optimize_result(
```

7. Model Evaluation

```
# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

In the confusion matrix, thee were 60,969 true negatives, 1,678 false negatives, 62,700 true positives and 2,249 false positives. The positives are low ride costs while the negatives are high ride costs.

This matrix reports a model accuracy of 96.92% (123,669 / 127,596)

```
# Report
print(classification_report(y_test, y_pred))
print("Accuracy:", round(accuracy_score(y_test, y_pred), 3))
```

```
              precision    recall  f1-score   support

       False       0.97      0.96      0.97     63218
        True       0.97      0.97      0.97     64378

    accuracy                           0.97    127596
   macro avg       0.97      0.97      0.97    127596
weighted avg       0.97      0.97      0.97    127596

Accuracy: 0.969
```

As for the classification report, it states what the confusion matrix had depicted. A overall near perfect model accuracy was accomplished in predicting whether the price of a ride from an Uber or Lyft in Bostom, MA would be Low or not.

Applying SMOTE due to imbalanced weather.value_counts()

```
# Apply smote
#!pip install imbalanced-learn
from imblearn.over_sampling import SMOTE
```

```
y.value_counts()
```

```
sm = SMOTE(random_state=42)
X_smote, y_smote = sm.fit_resample(X, y)
```

```
# After SMOTE
y_smote.value_counts()
```

```
X_smote.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 643088 entries, 0 to 643087
Data columns (total 64 columns):
```

Data columns (total 54 columns):

#	Column	Non-Null Count	Dtype
0	hour	643088 non-null	int64
1	day	643088 non-null	int64
2	month	643088 non-null	int64
3	distance	643088 non-null	float64
4	surge_multiplier	643088 non-null	float64
5	temperature	643088 non-null	float64
6	precipIntensity	643088 non-null	float64
7	precipProbability	643088 non-null	float64
8	humidity	643088 non-null	float64
9	windSpeed	643088 non-null	float64
10	windGust	643088 non-null	float64
11	visibility	643088 non-null	float64
12	temperatureHigh	643088 non-null	float64
13	temperatureLow	643088 non-null	float64
14	dewPoint	643088 non-null	float64
15	pressure	643088 non-null	float64
16	windBearing	643088 non-null	int64
17	cloudCover	643088 non-null	float64
18	uvIndex	643088 non-null	int64
19	ozone	643088 non-null	float64
20	moonPhase	643088 non-null	float64
21	precipIntensityMax	643088 non-null	float64
22	source_Beacon Hill	643088 non-null	bool
23	source_Boston University	643088 non-null	bool
24	source_Fenway	643088 non-null	bool
25	source_Financial District	643088 non-null	bool
26	source_Haymarket Square	643088 non-null	bool
27	source_North End	643088 non-null	bool
28	source_North Station	643088 non-null	bool
29	source_Northeastern University	643088 non-null	bool
30	source_South Station	643088 non-null	bool
31	source_Theatre District	643088 non-null	bool
32	source_West End	643088 non-null	bool
33	destination_Beacon Hill	643088 non-null	bool
34	destination_Boston University	643088 non-null	bool
35	destination_Fenway	643088 non-null	bool
36	destination_Financial District	643088 non-null	bool
37	destination_Haymarket Square	643088 non-null	bool
38	destination_North End	643088 non-null	bool
39	destination_North Station	643088 non-null	bool
40	destination_Northeastern University	643088 non-null	bool
41	destination_South Station	643088 non-null	bool
42	destination_Theatre District	643088 non-null	bool
43	destination_West End	643088 non-null	bool
44	cab_type_Uber	643088 non-null	bool
45	name_Black SUV	643088 non-null	bool
46	name_Lux	643088 non-null	bool
47	name_Lux Black	643088 non-null	bool
48	name_Lux Black XL	643088 non-null	bool
49	name_Lyft	643088 non-null	bool
50	name_Lyft XL	643088 non-null	bool
51	name_Shared	643088 non-null	bool
52	name_UberPool	643088 non-null	bool

```
## 4. Train-Test Split
X = df.drop(['price_cat_Low','price'], axis = 1)
y = df['price_cat_Low']
x_train, X_test, y_train, y_test = train_test_split(X_smote, y_smote, test_s
```

✦ Gemini

```
## 6. Train Logistic Regression Model

# Scale the data
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
X_test_scaled = scaler.transform(X_test)

model = LogisticRegression(max_iter=10000)
model.fit(x_train, y_train)
y_pred = model.predict(X_test)
model.fit(x_train_scaled, y_train)
y_pred = model.predict(X_test_scaled)
```

```
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_logistic.py:465
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
  n_iter_i = _check_optimize_result(
```

```
## 7. Model Evaluation
# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

```
# Report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

       False       0.97      0.96      0.97     64270
        True       0.96      0.97      0.97     64348

    accuracy                           0.97    128618
   macro avg       0.97      0.97      0.97    128618
weighted avg       0.97      0.97      0.97    128618
```

Same accuracy as done before without SMOTE.

Example: Predict whether the cost of a uber/lyft ride is high or low

Situation: Predict the cost of a Lyft on October 7th, at 9 am with a distance of 1.76 miles from North Station to South Station and a surge multiplier of 1.5. It is 60 F outside with rain, precipitation intensity of 0.125, precipitation probability of 1, humidity of 92%, wind speed of 10 mph, wind gust of 15 mph, visibility of 5 miles, a max temp of 70 F, low temp of 55 F, dew point of 57 F, pressure is 1000, wind bearing of 95, cloud cover of 100%, uv index 0, ozone of 285, moon phase of 0.7, and max precipitation of 0.15.

Model prediction parameters

Time & Charge: Hour, day, month, distance, surge multiplier:

[9, 7, 10, 1.76, 1.5]

[9, 7, 10, 1.76, 1.5]

Weather Details: temp, precipIntensity, precipProbability, humidity, windSpeed, windGust, visibility, tempHigh, tempLow, dewPoint, pressure, windBearing, cloudCover, uvIndex, ozone, moonPhase, precipIntensityMax

[60, 0.125, 1, 0.92, 10, 15, 5, 70, 55, 57, 1000, 95, 1, 0, 285, 0.7, 0.15]

Staring Location: source_Beacon Hill, source_Boston University, source_Fenway, source_Financial District, source_Haymarket Square, source_North End, source_North Station, source_Northeastern University, source_South Station, source_Theatre District, source_West End

[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]

Destination Location: destination_Beacon Hill, destination_Boston University, destination, destination_Financial District, destination_Haymarket Square, destination_North End, destination_North Station, destination_Northeastern University, destination_South Station, destination_Theatre District, destination_West End

[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]

Uber or Lyft and Vehicle Type: cab_type_Uber, name_Black SUV, name_Lux, name_Lux Black, name_Lux Black XL, name_Lyft, name_Lyft XL, name_Shared, name_UberPool, name_UberX, name_UberXL, name_WAV

[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]

Weather Type: weather_ Drizzle, weather_ Foggy, weather_ Light Rain, weather_ Mostly Cloudy, weather_ Overcast, weather_ Partly Cloudy, weather_ Possible Drizzle, weather_ Rain

[0, 0, 0, 0, 0, 0, 0, 1]

```
situation = [[9, 7, 10, 1.76, 1.5, 60, 0.125, 1, 0.92, 10, 15, 5, 70, 55, 57,
result = model.predict(situation)
print("The predicted ride cost is Low: ", result[0])
```

```
The predicted ride cost is Low:  True
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:2739: Use
  warnings.warn(
```

Overall, the model did very well predicting whether a ride cost in Boston, MA would be Low or not, achieving a 96.92% accuracy. It is surprising to learn that the weather doesn't influence the sales of Uber/Lyft rides. Only a minority of rides were ordered with rainy weather (the opposite was expected). My main finding is that weather

with rainy weather (the opposite was expected). My main finding is that weather doesn't imapct ride sales.