```
from IPython.display import Image
Image(url= "Uber_image.jpg")
```

```
import pandas as pd

# Use the 'quoting=3' argument to tell pandas to ignore quotes
uber_dataset = pd.read_csv("/content/rideshare_kaggle.csv", on_bad_lines='skip', quoting=3)
```

```
<ipython-input-121-724bcbb6f001>:4: DtypeWarning: Columns (18) have mixed types. Specify dtype option on import or set low_memory=Fa
  uber_dataset = pd.read_csv("/content/rideshare_kaggle.csv", on_bad_lines='skip', quoting=3)
```

```
uber_dataset.head()
```

	id	timestamp	hour	day	month	datetime	timezone	source	destination	cab_type	...	precipIntensityMax
0	424553bb-7174-41ea-aeb4-fe06d4f4b9d7	1.544953e+09	9	16	12	2018-12-16 09:30:07	America/New_York	Haymarket Square	North Station	Lyft	...	0.1276
1	4bd23055-6827-41c6-b23b-3c491f24e74d	1.543284e+09	2	27	11	2018-11-27 02:00:23	America/New_York	Haymarket Square	North Station	Lyft	...	0.1300
2	981a3613-77af-4620-a42a-0c0866077d1e	1.543367e+09	1	28	11	2018-11-28 01:00:22	America/New_York	Haymarket Square	North Station	Lyft	...	0.1064
3	c2d88af2-d278-4bfd-a8d0-29ca77cc5512	1.543554e+09	4	30	11	2018-11-30 04:53:02	America/New_York	Haymarket Square	North Station	Lyft	...	0.0000
4	e0126e1f-8ca9-4f2e-82b3-50505a09db9a	1.543463e+09	3	29	11	2018-11-29 03:49:20	America/New_York	Haymarket Square	North Station	Lyft	...	0.0001

5 rows × 57 columns

```
uber_dataset.shape
```

```
(45471, 57)
```

```
uber_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45471 entries, 0 to 45470
Data columns (total 57 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   id                           45471 non-null  object
 1   timestamp                    45471 non-null  float64
 2   hour                         45471 non-null  int64
 3   day                          45471 non-null  int64
 4   month                        45471 non-null  int64
 5   datetime                     45470 non-null  object
 6   timezone                     45470 non-null  object
 7   source                       45470 non-null  object
 8   destination                  45470 non-null  object
 9   cab_type                     45470 non-null  object
 10  product_id                   45470 non-null  object
 11  name                         45470 non-null  object
 12  price                        41911 non-null  float64
 13  distance                     45470 non-null  float64
 14  surge_multiplier             45470 non-null  float64
 15  latitude                     45470 non-null  float64
 16  longitude                    45470 non-null  float64
 17  temperature                  45470 non-null  float64
 18  apparentTemperature          45470 non-null  float64
 19  short_summary                45470 non-null  object
 20  long_summary                 45470 non-null  object
 21  precipIntensity              45470 non-null  float64
 22  precipProbability            45470 non-null  float64
 23  humidity                     45470 non-null  float64
 24  windSpeed                    45470 non-null  float64
 25  windGust                     45470 non-null  float64
 26  windGustTime                 45470 non-null  float64
```

```
 27  visibility                   45470 non-null  float64
 28  temperatureHigh              45470 non-null  float64
 29  temperatureHighTime          45469 non-null  float64
 30  temperatureLow               45469 non-null  float64
 31  temperatureLowTime           45469 non-null  float64
 32  apparentTemperatureHigh      45469 non-null  float64
 33  apparentTemperatureHighTime  45469 non-null  float64
 34  apparentTemperatureLow       45469 non-null  float64
 35  apparentTemperatureLowTime   45469 non-null  float64
 36  icon                         45469 non-null  object
 37  dewPoint                     45469 non-null  float64
 38  pressure                     45469 non-null  float64
 39  windBearing                  45469 non-null  float64
 40  cloudCover                   45469 non-null  float64
 41  uvIndex                      45469 non-null  float64
 42  visibility.1                 45469 non-null  float64
 43  ozone                        45469 non-null  float64
 44  sunriseTime                  45469 non-null  float64
 45  sunsetTime                   45469 non-null  float64
 46  moonPhase                    45469 non-null  float64
 47  precipIntensityMax           45469 non-null  float64
 48  uvIndexTime                  45469 non-null  float64
 49  temperatureMin               45469 non-null  float64
 50  temperatureMinTime           45469 non-null  float64
 51  temperatureMax               45469 non-null  float64
 52  temperatureMaxTime           45469 non-null  float64
```

```
uber_dataset.describe()
```

	timestamp	hour	day	month	price	distance	surge_multiplier	latitude	longitude
count	4.547100e+04	45471.000000	45471.000000	45471.000000	41911.000000	45470.000000	45470.000000	45470.000000	4.547000e+0
mean	1.544028e+09	11.578457	18.028986	11.572101	16.532328	2.181718	1.015441	42.343671	3.390275e+0
std	6.869970e+05	6.989026	9.987613	0.494780	9.312955	1.139033	0.104217	1.157732	7.244473e+0
min	1.543204e+09	0.000000	1.000000	11.000000	0.390000	0.000000	1.000000	42.214800	-7.110540e+0
25%	1.543439e+09	5.000000	13.000000	11.000000	9.000000	1.270000	1.000000	42.350300	-7.108100e+0
50%	1.543720e+09	12.000000	17.000000	12.000000	13.500000	2.140000	1.000000	42.351900	-7.106310e+0
75%	1.544814e+09	18.000000	28.000000	12.000000	22.500000	2.920000	1.000000	42.364700	-7.105420e+0
max	1.545161e+09	23.000000	30.000000	12.000000	92.000000	7.460000	9.874000	289.000000	1.544789e+0

8 rows × 46 columns

```
uber_dataset.isnull().sum()
```

	0
id	0
timestamp	0
hour	0
day	0
month	0
datetime	1
timezone	1
source	1
destination	1
cab_type	1
product_id	1
name	1
price	3560
distance	1
surge_multiplier	1
latitude	1
longitude	1
temperature	1
apparentTemperature	1
short_summary	1
long_summary	1
precipIntensity	1
precipProbability	1
humidity	1
windSpeed	1
windGust	1
windGustTime	1
visibility	1
temperatureHigh	1
temperatureHighTime	2
temperatureLow	2
temperatureLowTime	2
apparentTemperatureHigh	2
apparentTemperatureHighTime	2
apparentTemperatureLow	2
apparentTemperatureLowTime	2
icon	2
dewPoint	2
pressure	2
windBearing	2
cloudCover	2
uvIndex	2
visibility.1	2
ozone	2
sunriseTime	2
sunsetTime	2
moonPhase	2
precipIntensityMax	2
uvIndexTime	2

temperatureMin	2
temperatureMinTime	2
temperatureMax	2
temperatureMaxTime	2
apparentTemperatureMin	2
apparentTemperatureMinTime	2
apparentTemperatureMax	2
apparentTemperatureMaxTime	2

dtype: int64

```
uber_dataset = uber_dataset.dropna()
```

```
uber_dataset.isnull().sum()
```

	0
id	0
timestamp	0
hour	0
day	0
month	0
datetime	0
timezone	0
source	0
destination	0
cab_type	0
product_id	0
name	0
price	0
distance	0
surge_multiplier	0
latitude	0
longitude	0
temperature	0
apparentTemperature	0
short_summary	0
long_summary	0
precipIntensity	0
precipProbability	0
humidity	0
windSpeed	0
windGust	0
windGustTime	0
visibility	0
temperatureHigh	0
temperatureHighTime	0
temperatureLow	0
temperatureLowTime	0
apparentTemperatureHigh	0
apparentTemperatureHighTime	0
apparentTemperatureLow	0
apparentTemperatureLowTime	0
icon	0
dewPoint	0
pressure	0
windBearing	0
cloudCover	0
uvIndex	0
visibility.1	0
ozone	0
sunriseTime	0
sunsetTime	0
moonPhase	0
precipIntensityMax	0
uvIndexTime	0

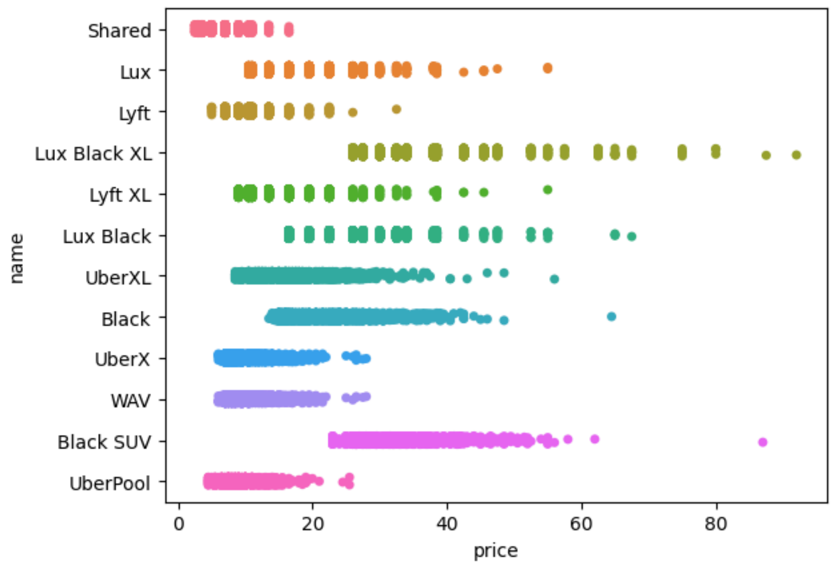
temperatureMin	0
temperatureMinTime	0
temperatureMax	0
temperatureMaxTime	0
apparentTemperatureMin	0
apparentTemperatureMinTime	0
apparentTemperatureMax	0
apparentTemperatureMaxTime	0

dtype: int64

```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
import seaborn as sns
import pandas as pd
```

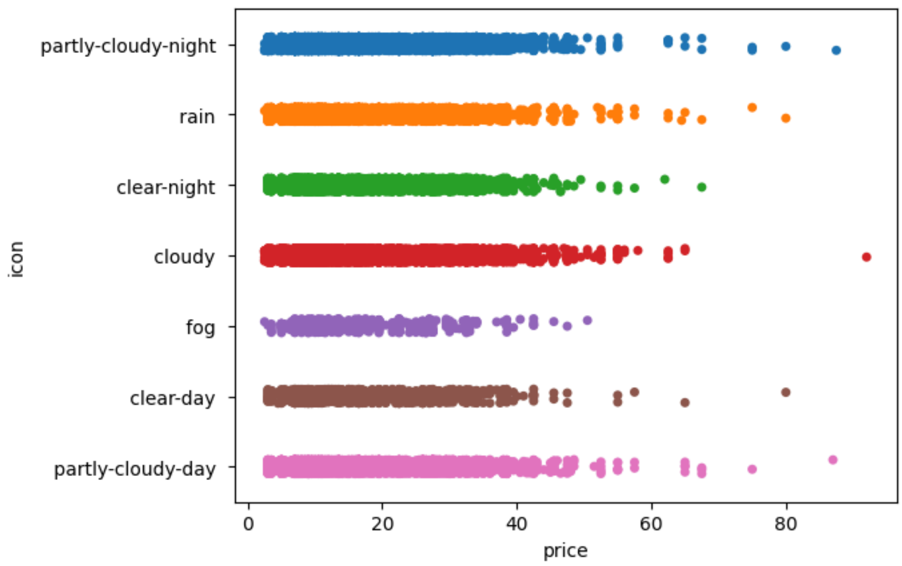
```
sns.stripplot(data=uber_dataset, x='price', y='name', hue='name')
```

```
<Axes: xlabel='price', ylabel='name'>
```



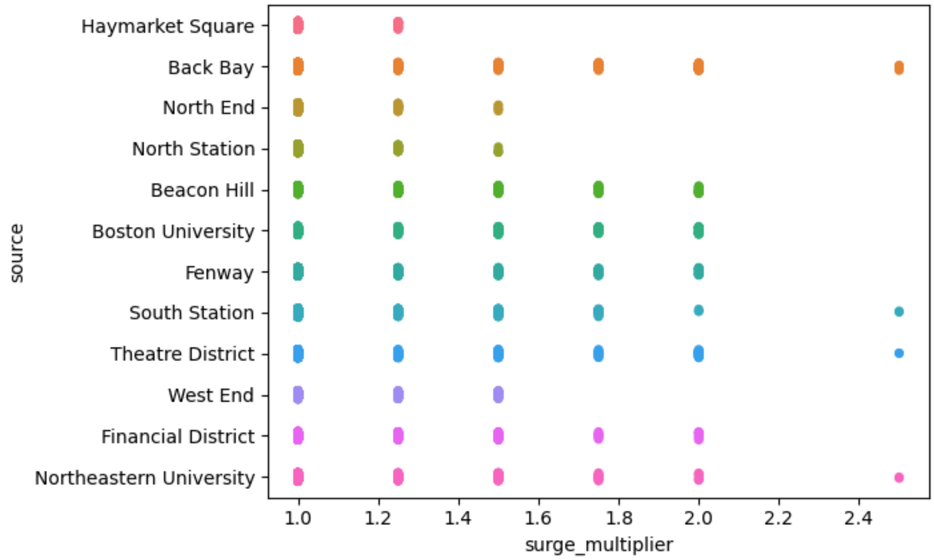

```
sns.stripplot(data=uber_dataset, x='price', y='icon', hue='icon')
```

```
<Axes: xlabel='price', ylabel='icon'>
```



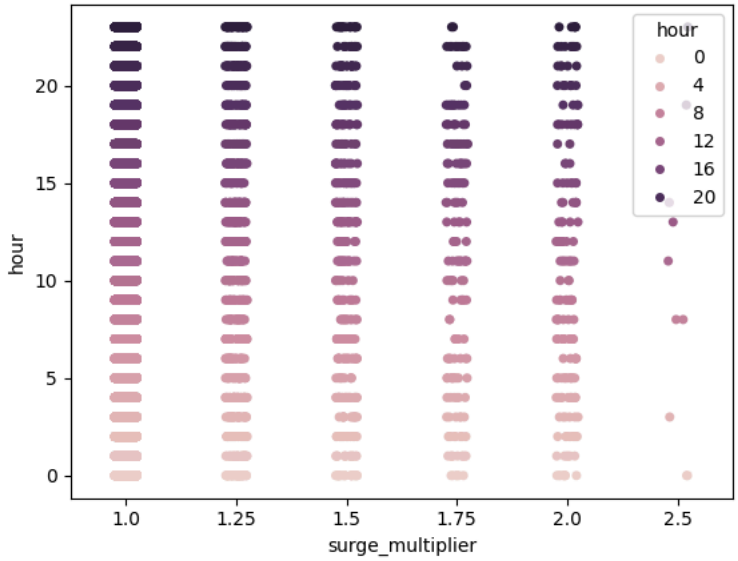

```
sns.stripplot(data=uber_dataset, x='surge_multiplier', y='source', hue='source')
```

<Axes: xlabel='surge_multiplier', ylabel='source'>




```
sns.stripplot(data=uber_dataset, x='surge_multiplier', y='hour', hue='hour')
```

<Axes: xlabel='surge_multiplier', ylabel='hour'>




```
uber_dataset['timestamp'].head()
```

	timestamp
0	1.544953e+09
1	1.543284e+09
2	1.543367e+09
3	1.543554e+09
4	1.543463e+09

dtype: float64

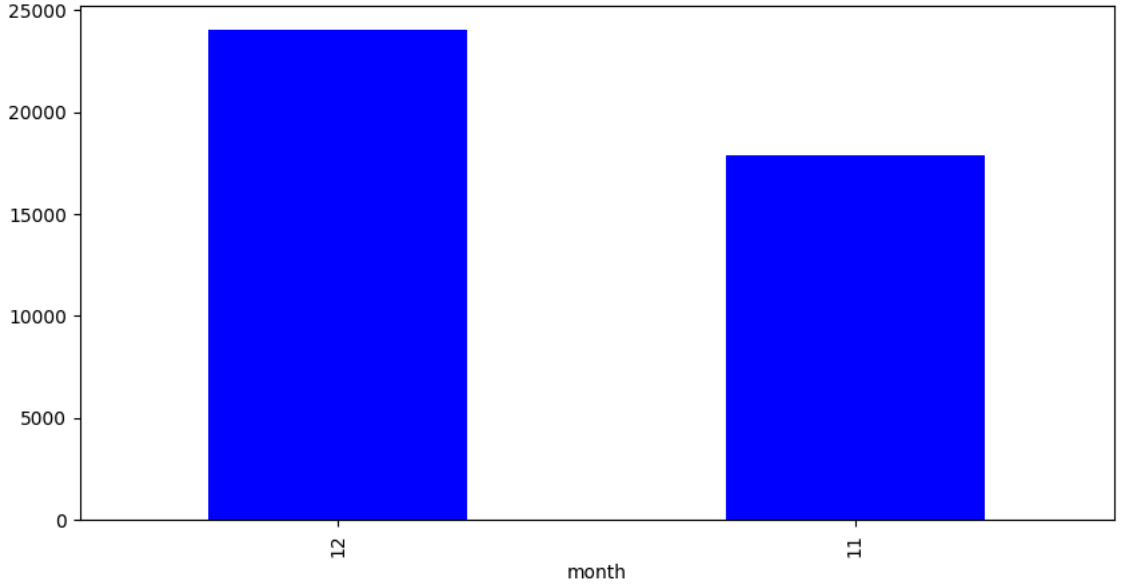
```
from datetime import datetime
timestamp1 = 1544952608
timestamp2 = 1543284024
timestamp3 = 1543818483
timestamp4 = 1543594384
timestamp5 = 1544728504
dt_object1 = datetime.fromtimestamp(timestamp1)
dt_object2 = datetime.fromtimestamp(timestamp2)
dt_object3 = datetime.fromtimestamp(timestamp3)
dt_object4 = datetime.fromtimestamp(timestamp4)
dt_object5 = datetime.fromtimestamp(timestamp5)

print("dt_object =", dt_object1)
print("dt_object =", dt_object2)
```

```
print("dt_object =", dt_object3)
print("dt_object =", dt_object4)
print("dt_object =", dt_object5)
```

```
dt_object = 2018-12-16 09:30:08
dt_object = 2018-11-27 02:00:24
dt_object = 2018-12-03 06:28:03
dt_object = 2018-11-30 16:13:04
dt_object = 2018-12-13 19:15:04
```

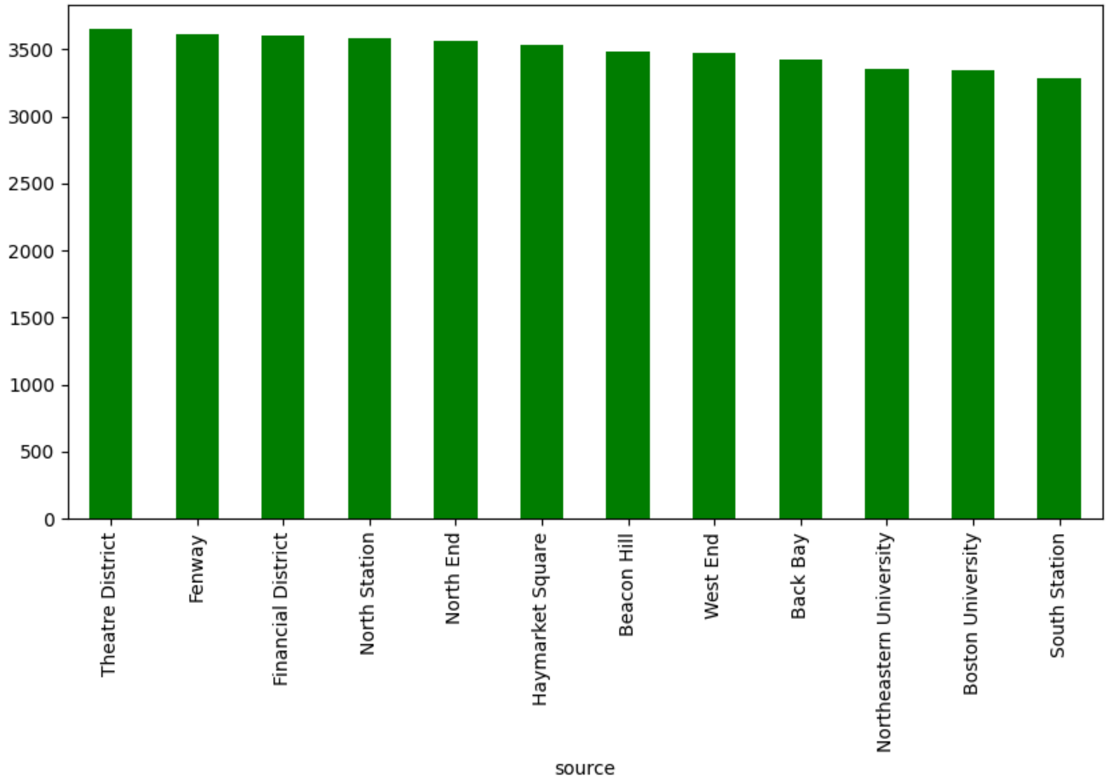
```
uber_dataset['month'].value_counts().plot(kind='bar', figsize=(10,5), color='blue')
```

```
<Axes: xlabel='month'>
```



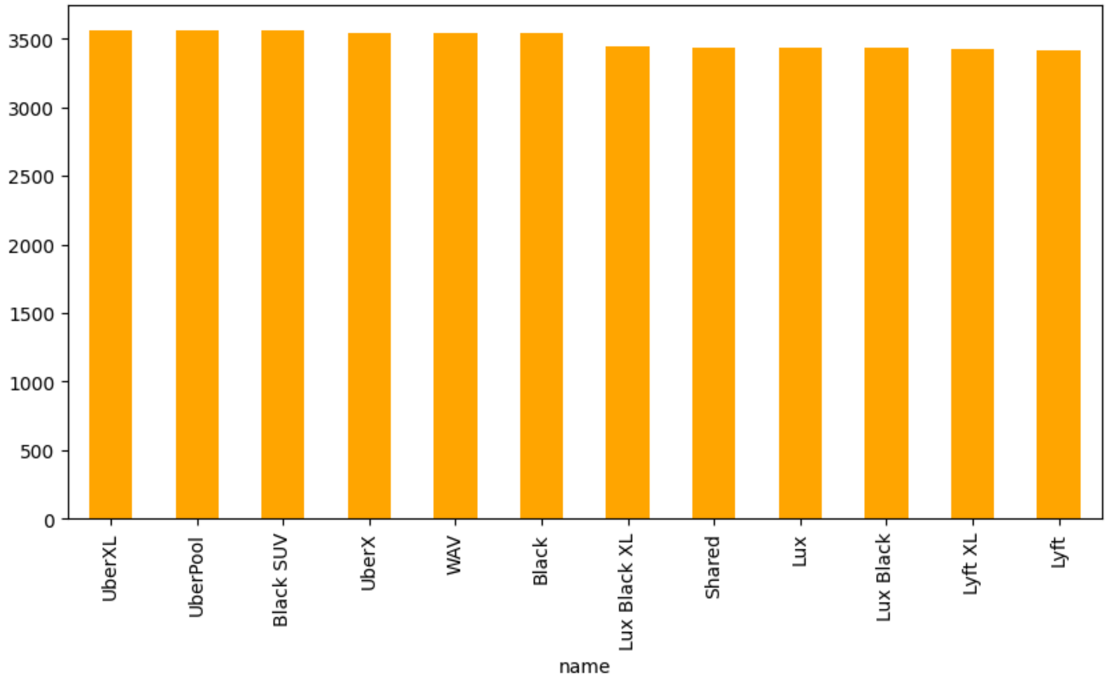

```
uber_dataset['source'].value_counts().plot(kind='bar', figsize=(10,5), color='green')
```

```
<Axes: xlabel='source'>
```



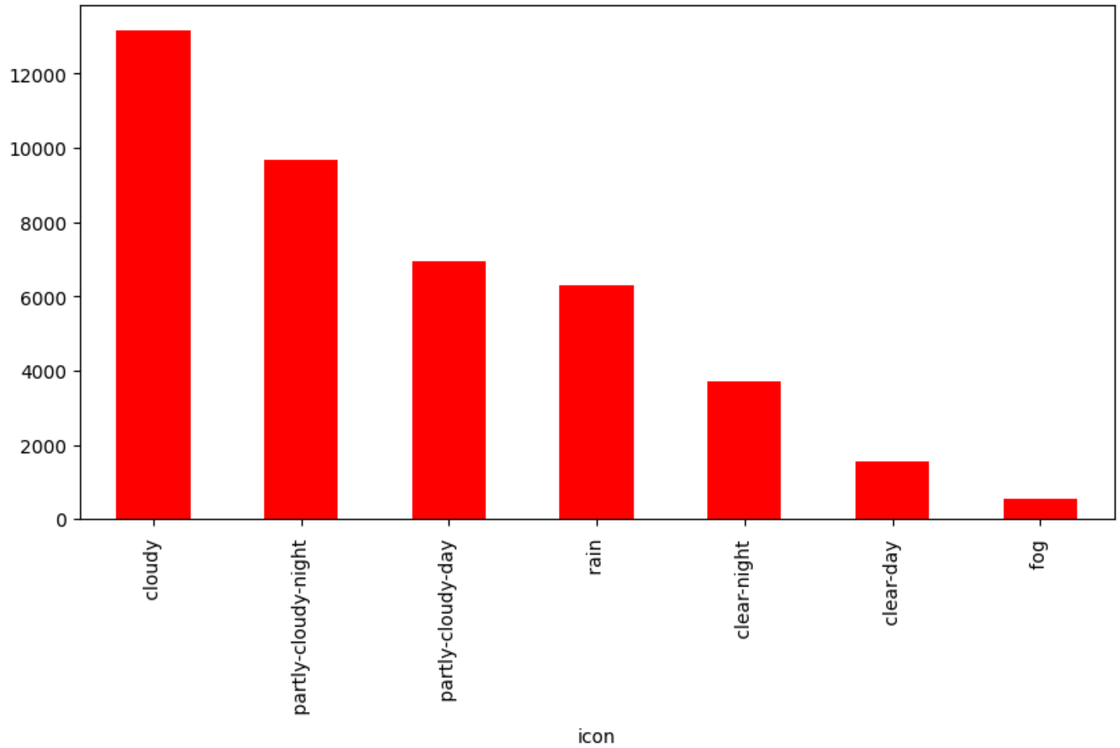

```
uber_dataset['name'].value_counts().plot(kind='bar', figsize=(10,5), color='orange')
```

```
<Axes: xlabel='name'>
```



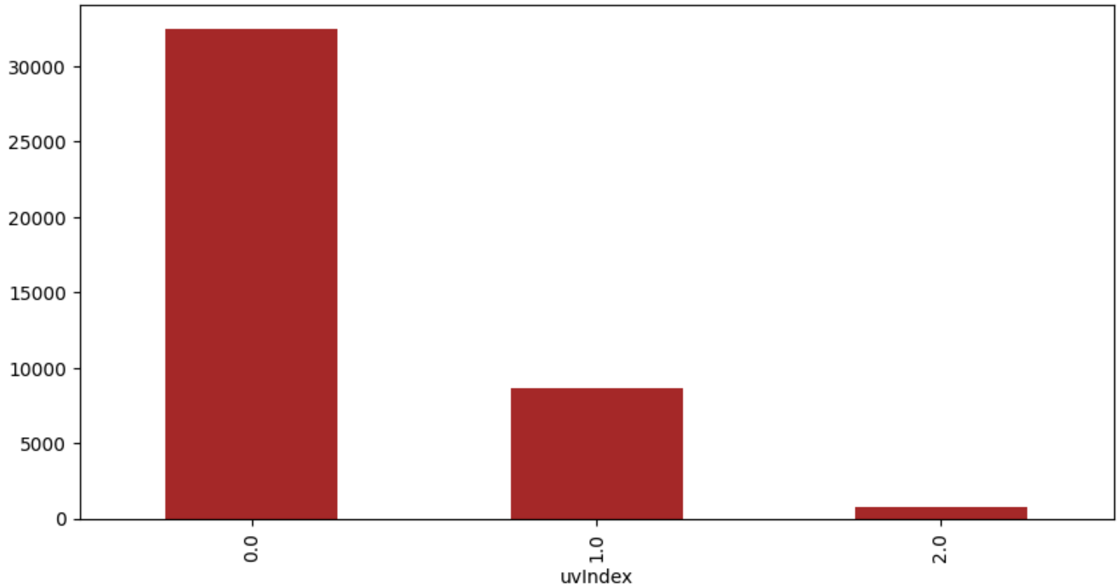

```
uber_dataset['icon'].value_counts().plot(kind='bar', figsize=(10,5), color='red')
```

```
<Axes: xlabel='icon'>
```



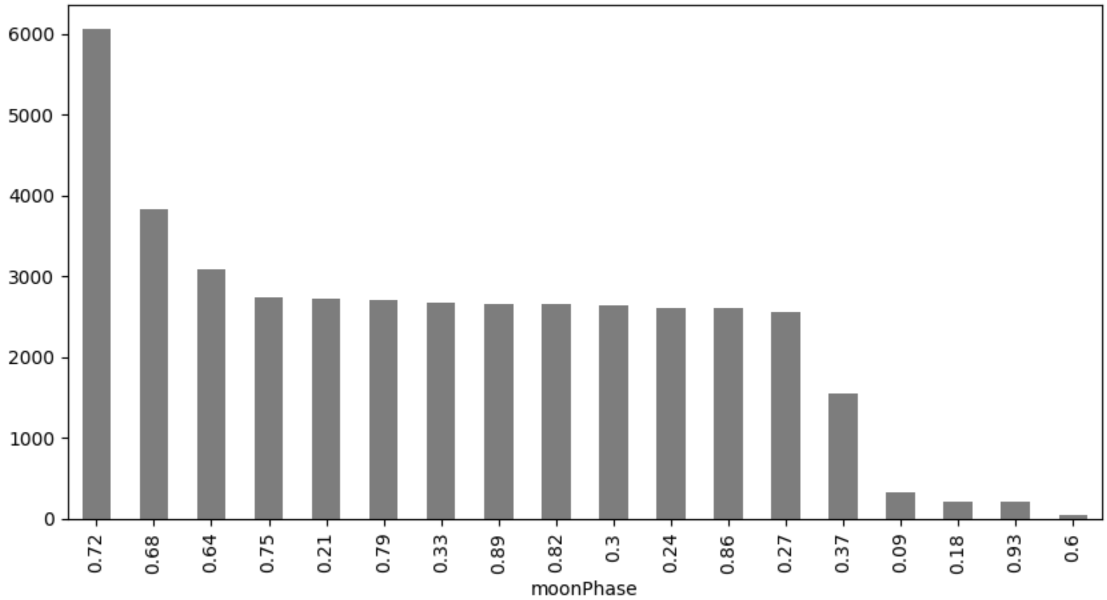

```
uber_dataset['uvIndex'].value_counts().plot(kind='bar', figsize=(10,5), color='brown')
```

```
<Axes: xlabel='uvIndex'>
```



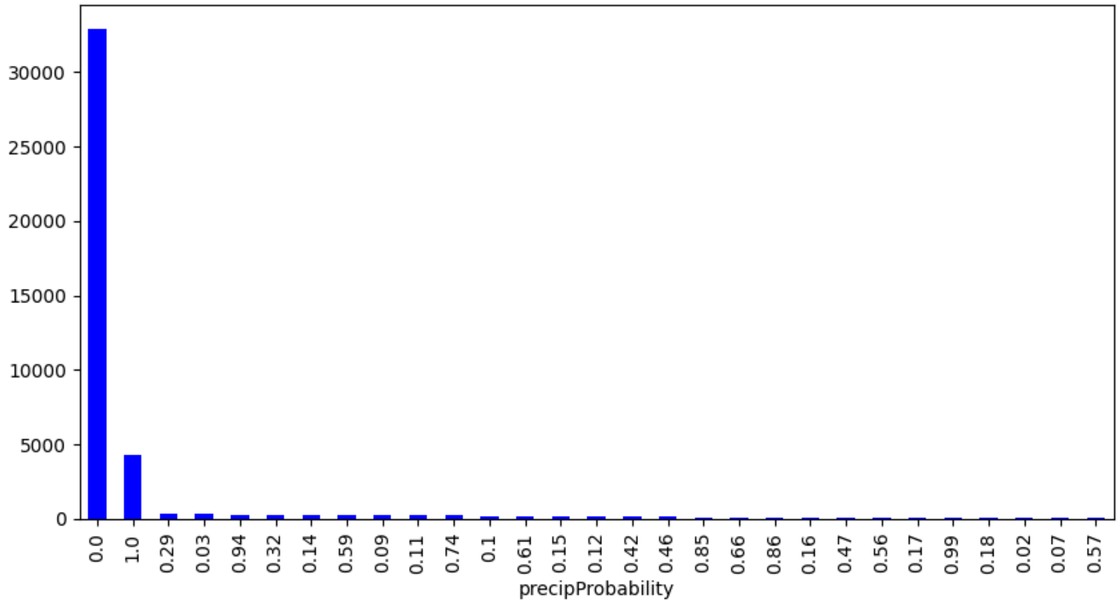

```
uber_dataset['moonPhase'].value_counts().plot(kind='bar', figsize=(10,5), color='grey')
```

```
<Axes: xlabel='moonPhase'>
```




```
uber_dataset['precipProbability'].value_counts().plot(kind='bar', figsize=(10,5), color='blue')
```

```
<Axes: xlabel='precipProbability'>
```




```
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()
```

```
uber_dataset.dtypes
```

	0
id	object
timestamp	float64
hour	int64
day	int64
month	int64
datetime	object
timezone	object
source	object
destination	object
cab_type	object
product_id	object
name	object
price	float64
distance	float64
surge_multiplier	float64
latitude	float64
longitude	float64
temperature	float64
apparentTemperature	float64
short_summary	object
long_summary	object
precipIntensity	float64
precipProbability	float64
humidity	float64
windSpeed	float64
windGust	float64
windGustTime	float64
visibility	float64
temperatureHigh	float64
temperatureHighTime	float64
temperatureLow	float64
temperatureLowTime	float64
apparentTemperatureHigh	float64
apparentTemperatureHighTime	float64
apparentTemperatureLow	float64
apparentTemperatureLowTime	float64
icon	object
dewPoint	float64
pressure	float64
windBearing	float64
cloudCover	float64
uvIndex	float64
visibility.1	float64
ozone	float64
sunriseTime	float64
sunsetTime	float64
moonPhase	float64
precipIntensityMax	float64
uvIndexTime	float64

temperatureMin	float64
temperatureMinTime	float64
temperatureMax	float64
temperatureMaxTime	float64
apparentTemperatureMin	float64
apparentTemperatureMinTime	float64
apparentTemperatureMax	float64
apparentTemperatureMaxTime	float64

```
uber_dataset['id']= label_encoder.fit_transform(uber_dataset['id'])
uber_dataset['datetime']= label_encoder.fit_transform(uber_dataset['datetime'])
uber_dataset['timezone']= label_encoder.fit_transform(uber_dataset['timezone'])
uber_dataset['destination']= label_encoder.fit_transform(uber_dataset['destination'])
uber_dataset['product_id']= label_encoder.fit_transform(uber_dataset['product_id'])
uber_dataset['short_summary']= label_encoder.fit_transform(uber_dataset['short_summary'])
uber_dataset['long_summary']= label_encoder.fit_transform(uber_dataset['long_summary'])
```

```
uber_dataset['name']= label_encoder.fit_transform(uber_dataset['name'])

print("Class mapping of Name: ")
for i, item in enumerate(label_encoder.classes_):
    print(item, "-->", i)
```

```
Class mapping of Name:
Black --> 0
Black SUV --> 1
Lux --> 2
Lux Black --> 3
Lux Black XL --> 4
Lyft --> 5
Lyft XL --> 6
Shared --> 7
UberPool --> 8
UberX --> 9
UberXL --> 10
WAV --> 11
```

```
uber_dataset['source']= label_encoder.fit_transform(uber_dataset['source'])

print("Class mapping of Source: ")
for i, item in enumerate(label_encoder.classes_):
    print(item, "-->", i)
```

```
Class mapping of Source:
Back Bay --> 0
Beacon Hill --> 1
Boston University --> 2
Fenway --> 3
Financial District --> 4
Haymarket Square --> 5
North End --> 6
North Station --> 7
Northeastern University --> 8
South Station --> 9
Theatre District --> 10
West End --> 11
```

```
uber_dataset['icon']= label_encoder.fit_transform(uber_dataset['icon'])

print("Class mapping of Icon: ")
for i, item in enumerate(label_encoder.classes_):
    print(item, "-->", i)
```

```
Class mapping of Icon:
 clear-day  --> 0
 clear-night  --> 1
 cloudy  --> 2
 fog  --> 3
 partly-cloudy-day  --> 4
 partly-cloudy-night  --> 5
 rain  --> 6
```

```
uber_dataset.dtypes
```

	0
id	int64
timestamp	float64
hour	int64
day	int64
month	int64
datetime	int64
timezone	int64
source	int64
destination	int64
cab_type	object
product_id	int64
name	int64
price	float64
distance	float64
surge_multiplier	float64
latitude	float64
longitude	float64
temperature	float64
apparentTemperature	float64
short_summary	int64
long_summary	int64
precipIntensity	float64
precipProbability	float64
humidity	float64
windSpeed	float64
windGust	float64
windGustTime	float64
visibility	float64
temperatureHigh	float64
temperatureHighTime	float64
temperatureLow	float64

```
uber_dataset.head()
```

	id	timestamp	hour	day	month	datetime	timezone	source	destination	cab_type	...	precipIntensityMax	uvIndexTime	te
0	10381	1.544953e+09	9	16	12	16884	0	5	7	Lyft	...	0.1276	1.544980e+09	
1	11813	1.543284e+09	2	27	11	772	0	5	7	Lyft	...	0.1300	1.543252e+09	
2	23784	1.543367e+09	1	28	11	2122	0	5	7	Lyft	...	0.1064	1.543338e+09	
3	30447	1.543554e+09	4	30	11	5364	0	5	7	Lyft	...	0.0000	1.543507e+09	
4	35036	1.543463e+09	3	29	11	3704	0	5	7	Lyft	...	0.0001	1.543421e+09	

5 rows × 57 columns

```
uber_dataset.isnull().sum()
```

	0
id	0
timestamp	0
hour	0
day	0
month	0
datetime	0
timezone	0
source	0
destination	0
cab_type	0
product_id	0
name	0
price	0
distance	0
surge_multiplier	0
latitude	0
longitude	0
temperature	0
apparentTemperature	0
short_summary	0
long_summary	0
precipIntensity	0
precipProbability	0
humidity	0
windSpeed	0
windGust	0
windGustTime	0
visibility	0
temperatureHigh	0
temperatureHighTime	0
temperatureLow	0
temperatureLowTime	0
apparentTemperatureHigh	0
apparentTemperatureHighTime	0
apparentTemperatureLow	0
apparentTemperatureLowTime	0
icon	0
dewPoint	0
pressure	0
windBearing	0
cloudCover	0
uvIndex	0
visibility.1	0
ozone	0
sunriseTime	0
sunsetTime	0
moonPhase	0
precipIntensityMax	0
uvIndexTime	0

temperatureMin	0
temperatureMinTime	0
temperatureMax	0
temperatureMaxTime	0
apparentTemperatureMin	0
apparentTemperatureMinTime	0
apparentTemperatureMax	0
apparentTemperatureMaxTime	0

```
uber_dataset['price'].median()
```

```
13.5
```

```
uber_dataset["price"].fillna(10.5, inplace = True)
```

```
uber_dataset.isnull().sum()
```

	0
id	0
timestamp	0
hour	0
day	0
month	0
datetime	0
timezone	0
source	0
destination	0
cab_type	0
product_id	0
name	0
price	0
distance	0
surge_multiplier	0
latitude	0
longitude	0
temperature	0
apparentTemperature	0
short_summary	0
long_summary	0
precipIntensity	0
precipProbability	0
humidity	0
windSpeed	0
windGust	0
windGustTime	0
visibility	0
temperatureHigh	0
temperatureHighTime	0
temperatureLow	0

```
uber_dataset['price'].dtype
```

```
dtype('float64')
```

apparentTemperatureLow 0

```
uber_dataset['price'] = uber_dataset['price'].astype(int)
```

```
uber_dataset['price'].head()
```

	price
0	5
1	11
2	7
3	26
4	9

sunsetTime 0

```
import numpy as np
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
```

uvIndexTime 0

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

temperatureMinTime 0

```
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor


from sklearn.feature_selection import RFE
```

apparentTemperatureMax 0

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

```
X.head()
```

	id	timestamp	hour	day	month	datetime	timezone	source	destination	cab_type	...	precipIntensityMax	uvIndexTime	te
0	10381	1.544953e+09	9	16	12	16884	0	5	7	Lyft	...	0.1276	1.544980e+09	
1	11813	1.543284e+09	2	27	11	772	0	5	7	Lyft	...	0.1300	1.543252e+09	
2	23784	1.543367e+09	1	28	11	2122	0	5	7	Lyft	...	0.1064	1.543338e+09	
3	30447	1.543554e+09	4	30	11	5364	0	5	7	Lyft	...	0.0000	1.543507e+09	
4	35036	1.543463e+09	3	29	11	3704	0	5	7	Lyft	...	0.0001	1.543421e+09	

5 rows × 56 columns

```
y.head()
```

	price
0	5
1	11
2	7
3	26
4	9

```
X.shape
```

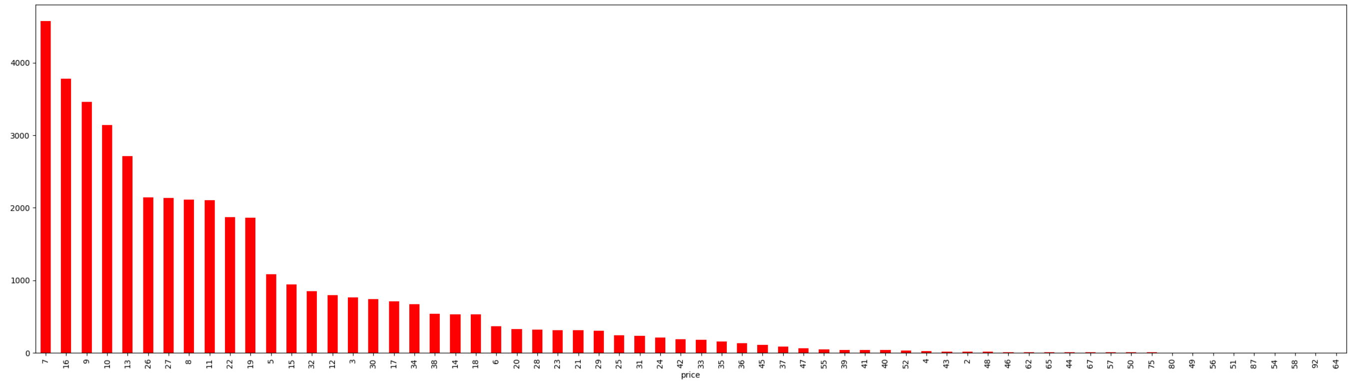
```
(41910, 56)
```

```
y.shape
```

```
(41910,)
```

```
y.value_counts().plot(kind='bar',figsize=(30,8),color='red')
```

```
<Axes: xlabel='price'>
```




```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

```
# Convert y_train to a numeric type, coercing errors to NaN
y_train = pd.to_numeric(y_train, errors='coerce')

# Drop rows with NaN values in y_train
X_train = X_train[y_train.notna()]
y_train = y_train[y_train.notna()]
```

```
X_train.shape
```

```
(33528, 56)
```

```
X_test.shape
```

```
(8382, 56)
```

```
y_train.shape
```

```
(33528,)
```

```
y_test.shape
```

```
(8382,)
```

```
from sklearn.linear_model import LinearRegression
```

```
model = LinearRegression()
```

```
# Check for columns with 'Lyft' in X_train
lyft_columns = [col for col in X_train.columns if X_train[col].astype(str).str.contains('Lyft').any()]

# Print the columns containing 'Lyft'
print(f"Columns containing 'Lyft': {lyft_columns}")

# Import LabelEncoder
from sklearn.preprocessing import LabelEncoder

# Apply label encoding to the identified columns
for col in lyft_columns:
    le = LabelEncoder()
    X_train[col] = le.fit_transform(X_train[col])

# Now you can try fitting the model again
model.fit(X_train, y_train)
```

```
Columns containing 'Lyft': ['cab_type']
▾ LinearRegression
LinearRegression()
```

```
model.score(X_train, y_train) # Use 'model' instead of 'reg' to call the score function.
```

```
0.544811801511188
```

```
# Check for columns with 'Lyft' in X
lyft_columns = [col for col in X.columns if X[col].astype(str).str.contains('Lyft').any()]

# Print the columns containing 'Lyft'
print(f"Columns containing 'Lyft': {lyft_columns}")

# Apply label encoding to the identified columns in X
for col in lyft_columns:
    le = LabelEncoder()
    X[col] = le.fit_transform(X[col])

# Now you can try fitting the model again
rfe = RFE(model, n_features_to_select=40, verbose=1) # Use 'model' instead of 'reg'
rfe = rfe.fit(X, y)
```

```
Columns containing 'Lyft': ['cab_type']
Fitting estimator with 56 features.
Fitting estimator with 55 features.
Fitting estimator with 54 features.
Fitting estimator with 53 features.
Fitting estimator with 52 features.
Fitting estimator with 51 features.
Fitting estimator with 50 features.
Fitting estimator with 49 features.
Fitting estimator with 48 features.
Fitting estimator with 47 features.
Fitting estimator with 46 features.
Fitting estimator with 45 features.
Fitting estimator with 44 features.
Fitting estimator with 43 features.
Fitting estimator with 42 features.
Fitting estimator with 41 features.
```

```
rfe.support_
```

```
array([False, False,  True,  True,  True, False, False,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True, False,  True,
        True, False,  True, False,  True, False,  True, False,  True,
        True,  True, False,  True,  True,  True,  True, False,  True,
        True,  True, False,  True, False,  True, False,  True, False,
        True, False])
```

```
XX = X[X.columns[rfe.support_]]
```

```
XX.head()
```

	hour	day	month	source	destination	cab_type	product_id	name	distance	surge_multiplier	...	uvIndex	visibility.1	ozone
0	9	16	12	5	7	0	7	7	0.44	1.0	...	0.0	10.000	303.8
1	2	27	11	5	7	0	11	2	0.44	1.0	...	0.0	4.786	291.1
2	1	28	11	5	7	0	6	5	0.44	1.0	...	0.0	10.000	315.7
3	4	30	11	5	7	0	9	4	0.44	1.0	...	0.0	10.000	291.1
4	3	29	11	5	7	0	10	6	0.44	1.0	...	0.0	10.000	347.7

5 rows × 40 columns

```
X_train, X_test, y_train, y_test = train_test_split(XX, y, test_size = 0.3, random_state = 10)
```

```
X_train.shape
```

```
(29337, 40)
```

```
#Creating model
reg1 = LinearRegression()
#Fitting training data
reg1 = reg1.fit(X_train, y_train)
```

```
reg1.score(X_train, y_train)
```

```
0.5427936128993119
```

```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression

# Create a LinearRegression object
```

```
reg = LinearRegression()

# Create the RFE object, specifying n_features_to_select as a keyword argument
rfe = RFE(reg, n_features_to_select = 15, verbose=1)

# Fit the RFE object to the data
rfe = rfe.fit(X, y)
```

```
Fitting estimator with 56 features.
Fitting estimator with 55 features.
Fitting estimator with 54 features.
Fitting estimator with 53 features.
Fitting estimator with 52 features.
Fitting estimator with 51 features.
Fitting estimator with 50 features.
Fitting estimator with 49 features.
Fitting estimator with 48 features.
Fitting estimator with 47 features.
Fitting estimator with 46 features.
Fitting estimator with 45 features.
Fitting estimator with 44 features.
Fitting estimator with 43 features.
Fitting estimator with 42 features.
Fitting estimator with 41 features.
Fitting estimator with 40 features.
Fitting estimator with 39 features.
Fitting estimator with 38 features.
Fitting estimator with 37 features.
Fitting estimator with 36 features.
Fitting estimator with 35 features.
Fitting estimator with 34 features.
Fitting estimator with 33 features.
Fitting estimator with 32 features.
Fitting estimator with 31 features.
Fitting estimator with 30 features.
Fitting estimator with 29 features.
Fitting estimator with 28 features.
Fitting estimator with 27 features.
Fitting estimator with 26 features.
Fitting estimator with 25 features.
Fitting estimator with 24 features.
Fitting estimator with 23 features.
Fitting estimator with 22 features.
Fitting estimator with 21 features.
Fitting estimator with 20 features.
Fitting estimator with 19 features.
Fitting estimator with 18 features.
Fitting estimator with 17 features.
Fitting estimator with 16 features.
```

```
XX = X[X.columns[rfe.support_]]
```

```
XX.head()
```

	cab_type	product_id	name	distance	surge_multiplier	latitude	longitude	precipIntensity	humidity	temperatureHigh	apparent
0	0	7	7	0.44	1.0	42.2148	-71.033	0.0000	0.68	43.68	
1	0	11	2	0.44	1.0	42.2148	-71.033	0.1299	0.94	47.30	
2	0	6	5	0.44	1.0	42.2148	-71.033	0.0000	0.75	47.55	
3	0	9	4	0.44	1.0	42.2148	-71.033	0.0000	0.73	45.03	
4	0	10	6	0.44	1.0	42.2148	-71.033	0.0000	0.70	42.18	

```
X_train, X_test, y_train, y_test = train_test_split(XX, y, test_size = 0.3, random_state = 10,)
```

```
X_train.shape
```

```
(29337, 15)
```

```
#Creating model
reg1 = LinearRegression()
#Fitting training data
reg1 = reg1.fit(X_train, y_train)
```

```
reg1.score(X_train, y_train)
```

```
0.542392692716986
```

```
rfe = RFE(reg1, n_features_to_select=25, verbose=1)
rfe = rfe.fit(X, y)
```

```
Fitting estimator with 56 features.
Fitting estimator with 55 features.
Fitting estimator with 54 features.
Fitting estimator with 53 features.
Fitting estimator with 52 features.
Fitting estimator with 51 features.
Fitting estimator with 50 features.
Fitting estimator with 49 features.
Fitting estimator with 48 features.
Fitting estimator with 47 features.
Fitting estimator with 46 features.
Fitting estimator with 45 features.
Fitting estimator with 44 features.
Fitting estimator with 43 features.
Fitting estimator with 42 features.
Fitting estimator with 41 features.
Fitting estimator with 40 features.
Fitting estimator with 39 features.
Fitting estimator with 38 features.
Fitting estimator with 37 features.
Fitting estimator with 36 features.
Fitting estimator with 35 features.
Fitting estimator with 34 features.
Fitting estimator with 33 features.
Fitting estimator with 32 features.
Fitting estimator with 31 features.
Fitting estimator with 30 features.
Fitting estimator with 29 features.
Fitting estimator with 28 features.
Fitting estimator with 27 features.
Fitting estimator with 26 features.
```

```
XX = X[X.columns[rfe.support_]]
```

```
XX.head()
```

	month	source	cab_type	product_id	name	distance	surge_multiplier	latitude	longitude	temperature	...	temperatureHigh	app
0	12	5	0	7	7	0.44	1.0	42.2148	-71.033	42.34	...	43.68	
1	11	5	0	11	2	0.44	1.0	42.2148	-71.033	43.58	...	47.30	
2	11	5	0	6	5	0.44	1.0	42.2148	-71.033	38.33	...	47.55	
3	11	5	0	9	4	0.44	1.0	42.2148	-71.033	34.38	...	45.03	
4	11	5	0	10	6	0.44	1.0	42.2148	-71.033	37.44	...	42.18	

5 rows × 25 columns

```
X_train, X_test, y_train, y_test = train_test_split(XX, y, test_size = 0.3, random_state = 20,)
```

```
X_train.shape
```

```
(29337, 25)
```

```
#Creating model
reg1 = LinearRegression()
#Fitting training data
reg1 = reg1.fit(X_train, y_train)
#Y prediction
Y_pred = reg1.predict(X_test)
```

```
reg1.score(X_train, y_train)
```

```
0.5424603939185252
```

```
XX.columns
```

```
Index(['month', 'source', 'cab_type', 'product_id', 'name', 'distance',
       'surge_multiplier', 'latitude', 'longitude', 'temperature',
       'apparentTemperature', 'precipIntensity', 'precipProbability',
       'humidity', 'windSpeed', 'temperatureHigh', 'apparentTemperatureHigh',
       'dewPoint', 'uvIndex', 'moonPhase', 'precipIntensityMax',
       'temperatureMin', 'temperatureMax', 'apparentTemperatureMin',
       'apparentTemperatureMax'],
      dtype='object')
```

```
XX.shape
```

(41910, 25)

```
XX.head()
```

	month	source	cab_type	product_id	name	distance	surge_multiplier	latitude	longitude	temperature	...	temperatureHigh	app
0	12	5	0	7	7	0.44	1.0	42.2148	-71.033	42.34	...	43.68	
1	11	5	0	11	2	0.44	1.0	42.2148	-71.033	43.58	...	47.30	
2	11	5	0	6	5	0.44	1.0	42.2148	-71.033	38.33	...	47.55	
3	11	5	0	9	4	0.44	1.0	42.2148	-71.033	34.38	...	45.03	
4	11	5	0	10	6	0.44	1.0	42.2148	-71.033	37.44	...	42.18	

5 rows × 25 columns

```
columns_to_drop = ['latitude', 'longitude', 'apparentTemperature', 'long_summary', 'precipIntensity',
                   'humidity', 'windSpeed', 'windGust', 'temperatureHigh', 'apparentTemperatureHigh',
                   'dewPoint', 'precipIntensityMax', 'temperatureMax', 'apparentTemperatureMax',
                   'distance', 'cloudCover', 'moonPhase']

# Filtering only existing columns to drop
columns_to_drop = [col for col in columns_to_drop if col in uber_dataset.columns]

# Dropping the columns
uber_dataset = uber_dataset.drop(columns=columns_to_drop)
uber_dataset= uber_dataset.drop(columns=columns_to_drop, errors='ignore')
```

```
uber_dataset.head()
```

	id	timestamp	hour	day	month	datetime	timezone	source	destination	cab_type	...	ozone	sunriseTime	sunsetTime	
0	10381	1.544953e+09	9	16	12	16884	0	5	7	Lyft	...	303.8	1.544962e+09	1.544995e+09	1
1	11813	1.543284e+09	2	27	11	772	0	5	7	Lyft	...	291.1	1.543233e+09	1.543267e+09	1
2	23784	1.543367e+09	1	28	11	2122	0	5	7	Lyft	...	315.7	1.543319e+09	1.543353e+09	1