US Cars Dataset

Online Car Auction in North American

US Cars
Dataset

US Cars'data was scraped from AUCTION EXPORT.com. This dataset included Information about 28 brands of clean and used vehicles for sale in US. Twelve features were assembled for each car in the dataset.

https://www.kaggle.com/doaaalsenani/usa-cers-dataset

Our Goals [/]

1

Give some insights about the US Cars Dataset.

Answer some questions that may jump up to your head.

2

Provide useful visualizations based on the data set.

Data visualizations is one of the best ways to understand the data.

3

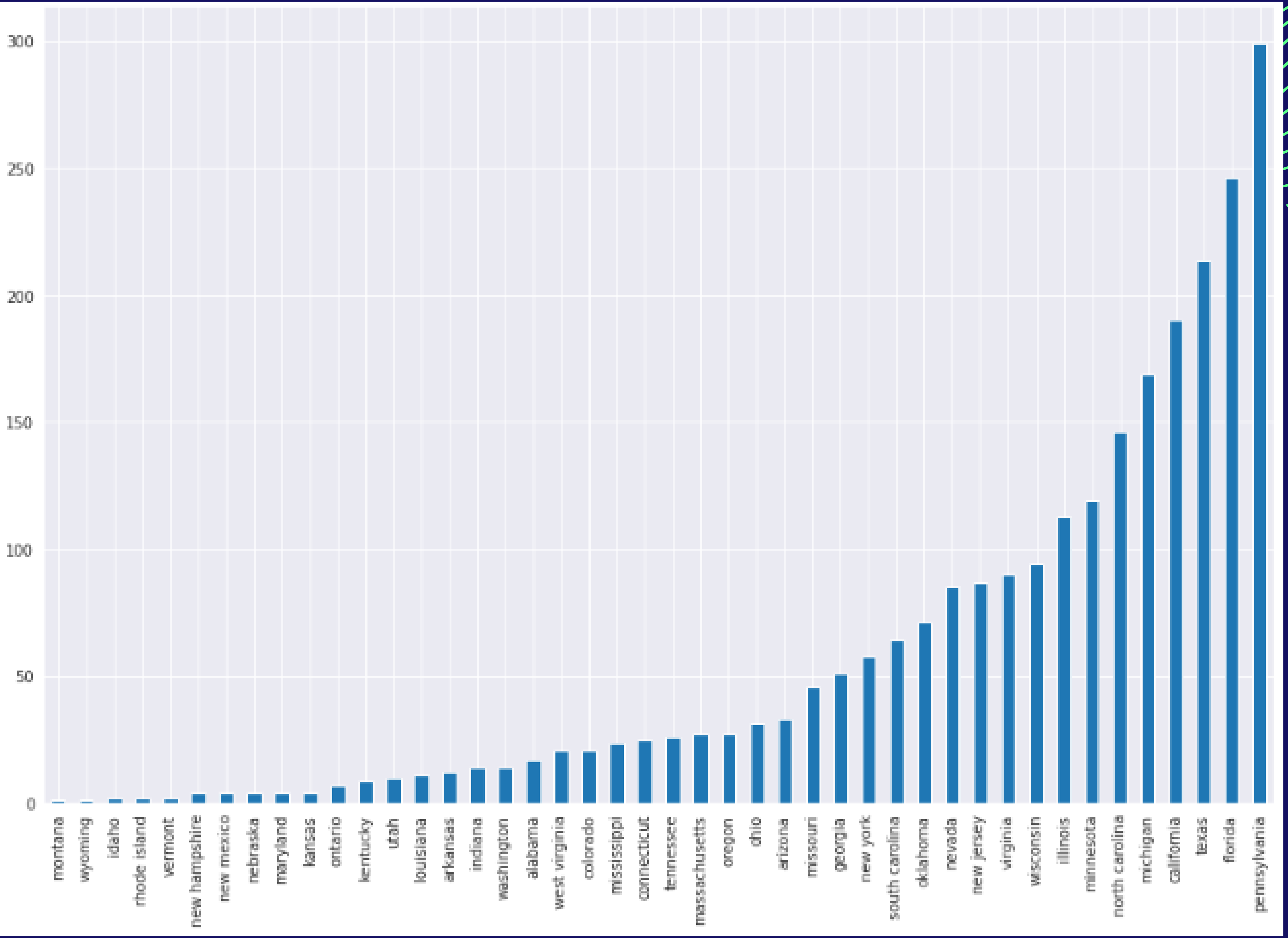
Train a machine learing model and calculate its accuracy.

I tested machine learning models to predict the price of the cars.

We have 12 features in our dataset

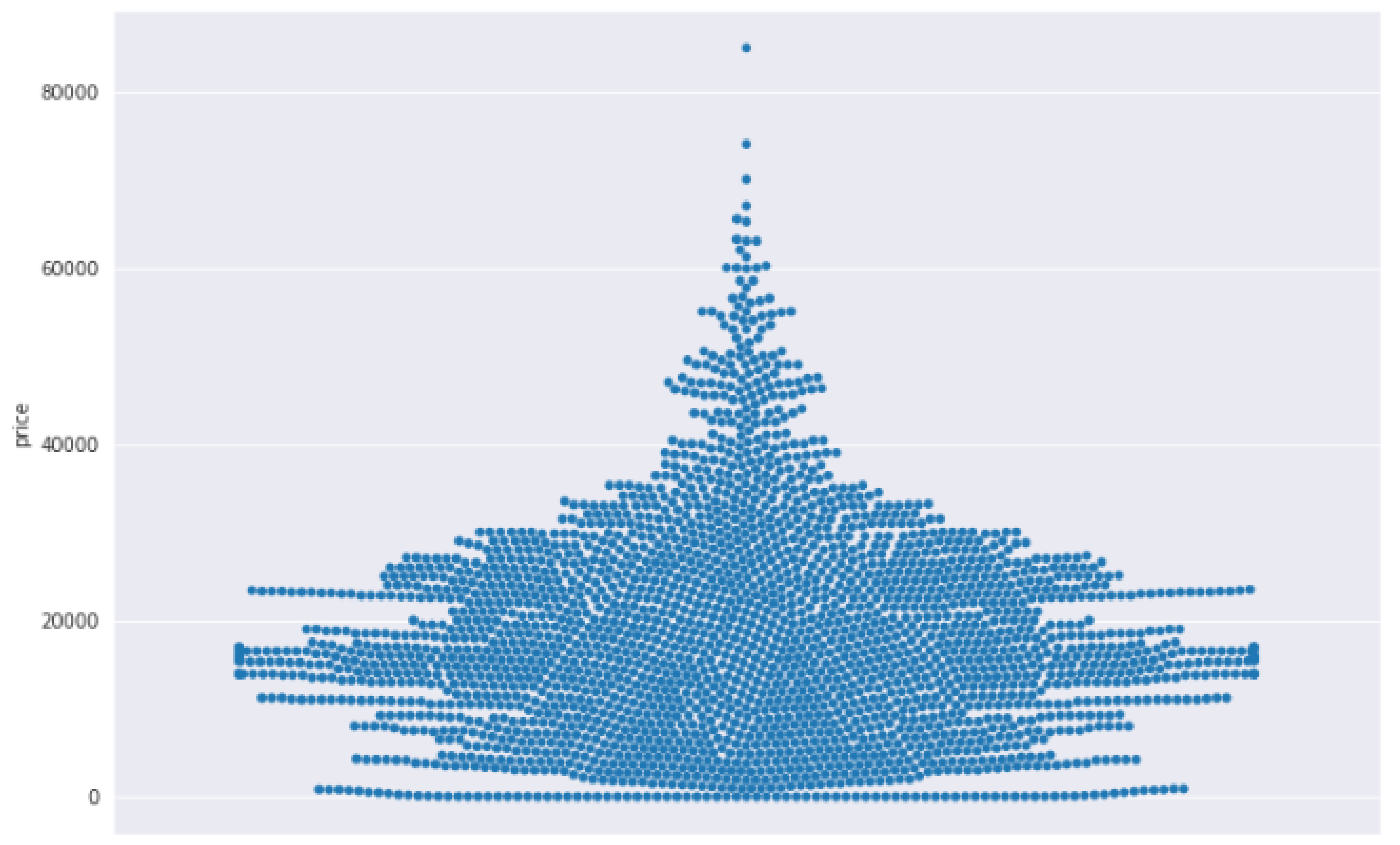
Feature	Type	Description
Price	Integer	The sale price of the vehicle in the ad
Years	Integer	The vehicle registration year
Brand	String	The brand of car
Model	String	model of the vehicle
Color	String	Color of the vehicle
State/City	String	The location in which the car is being available for purchase
Mileage	Float	miles traveled by vehicle
Vin	String	The vehicle identification number is a collection of 17 characters (digits and capital letters)
Title Status	String	This feature included binary classification, which are clean title vehicles and salvage insurance
Lot	Integer	A lot number is an identification number assigned to a particular quantity or lot of material from a single manufacturer.For cars, a lot number is combined with a serial number to form the Vehicle Identification Number.
Condition	String	Time

Data Visualization



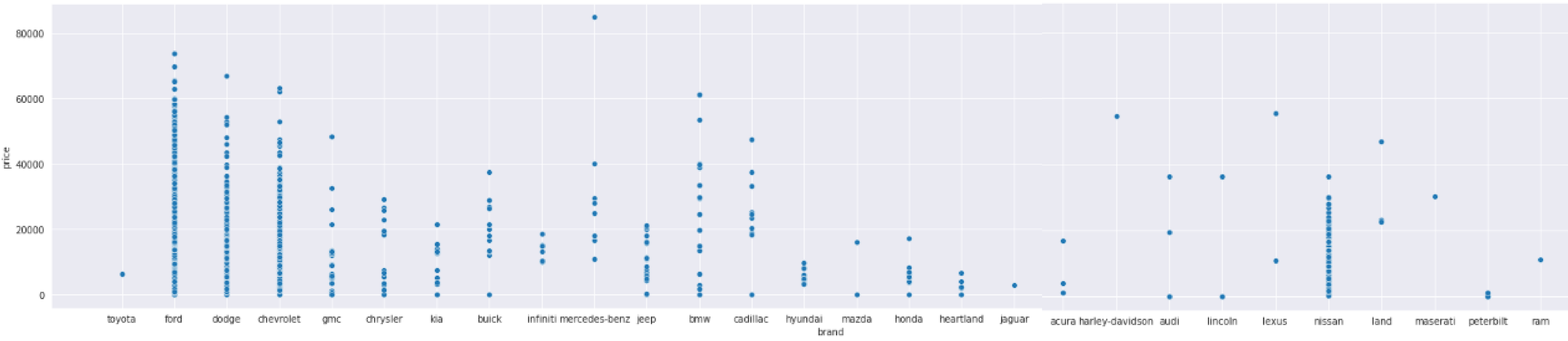
Number of cars for each state.

Data Visualization



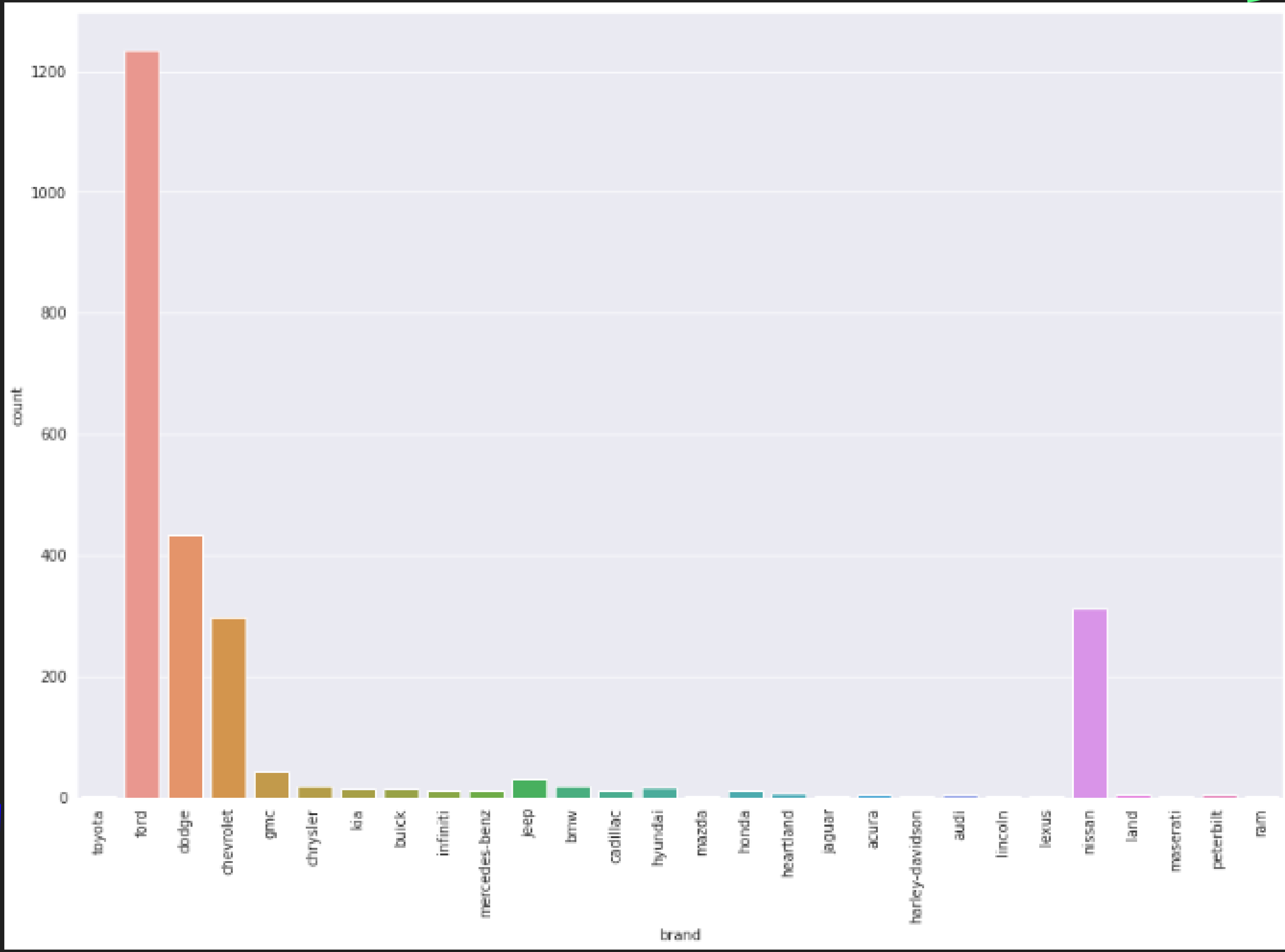
The flow of the price

Data Visualization



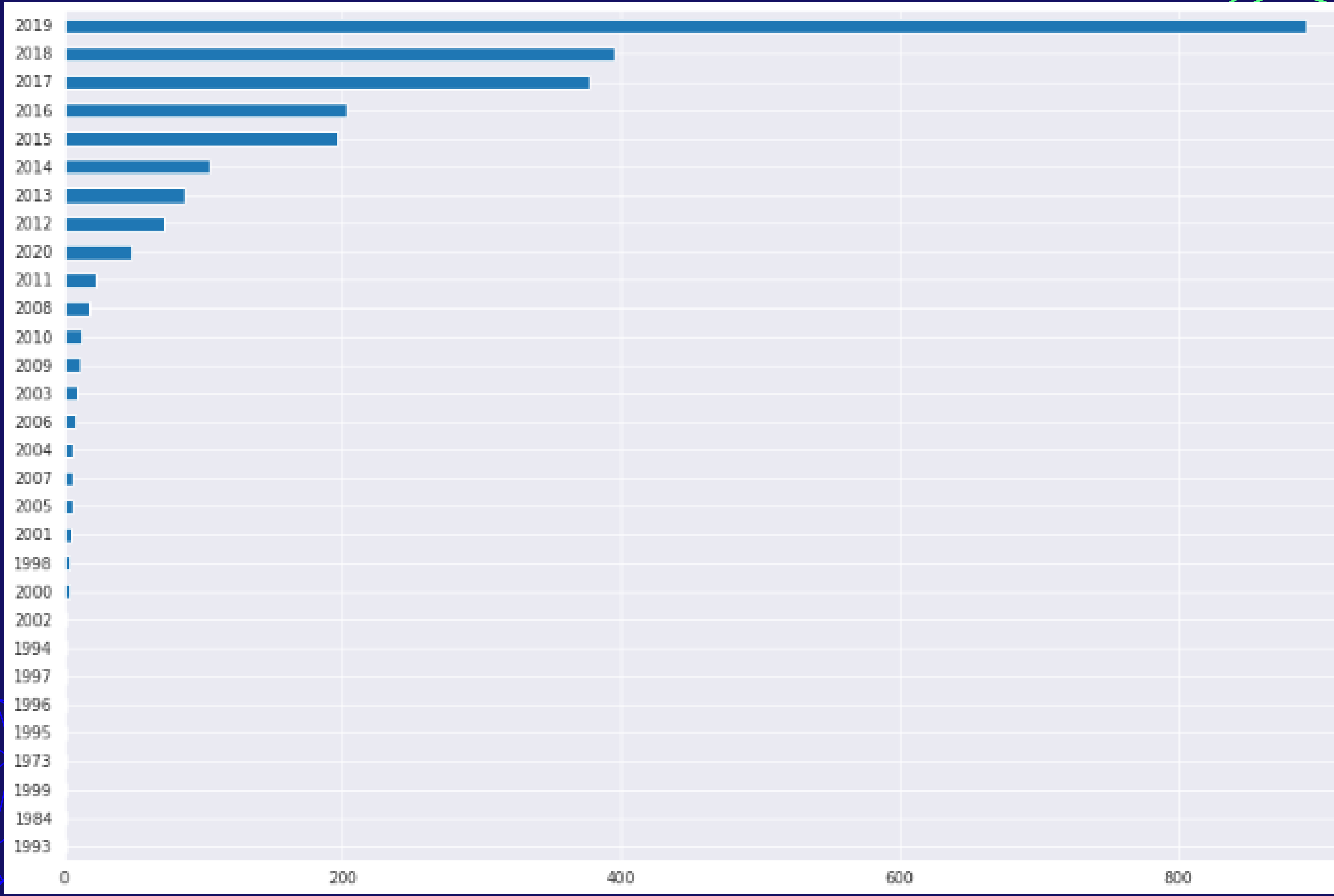
Distribution of prices for each brand.

Data Visualization



Number of cars for each brand.

Data Visualization



Number of cars for each year.

Our Data Insights.

Answer some of questions that may jump up to your head.

- Which brand has the highest price?
- When was the most expensive Auction?
- Is the color of the car affected by its registration year in the 21st century?
- What's the propability that a FORD car has a price more than the avarage price for all cars?
- Which model has the lowest mileage?
- Which state has the highest number of DODGEs?
- What's the most popular color in the cheapest cars?
- Which is the best selling brand in Virginia?
- When was ford the best selling car?

What's the most popular color in the cheapest model?

```
#What's the most popular color in the cheapest cars?
df[df['price'] == df['price'].min()]['color'].value_counts()#.
#Black, Grey, and Green
```

black	6
gray	6
green	6
white	5
red	5
silver	4
blue	4
orange	2
gold	2
maroon	1
yellow	1
light blue	1

Answer:
Black, Grey, and Green.

- Which brand has the highest price?
- When was the most expensive Auction?

```
[50] df.sort_values("price").tail()
```

	price	brand	model	year	title_status	mileage	color	state	country	brand_num
1215	65500	ford	srw	2019	clean vehicle	6500.0	black	indiana	usa	8
277	67000	dodge	challenger	2019	clean vehicle	10944.0	blue	ohio	usa	7
1336	70000	ford	drw	2019	clean vehicle	9643.0	no_color	illinois	usa	8
1340	74000	ford	drw	2019	clean vehicle	10536.0	no_color	illinois	usa	8
502	84900	mercedes-benz	sl-class	2017	clean vehicle	25302.0	silver	florida	usa	23

1. Mercedes-benz
2. 2017

Is the color of the car affected by its registration year in the 21st century?

```
#Is the color of the car affected by its registration year in the 21st century (we have a little data about the 20th centry)?
lst1 = []
lst2 = []
for year in df[df['year'] > 1999]['year'].sort_values().unique():
    x = df[df['year'] == year].value_counts('color')[0]     #Number of cars whose color is dominant for this particular year.
    print(year, df[df['year'] == year].value_counts('color').idxmax(), x)     #Print year, dominant color, and number of cars of this color.
    lst1.append(year)
    lst2.append(df[df['year'] == year].value_counts('color').idxmax())
yc = pd.DataFrame(list(zip(lst1, lst2)), columns =['year', 'color'])
print (yc['color'].value_counts())     #Number of years when each color was dominant.
#yes, Although White was the dominant color in 12 years, the dominance wasn't complete.
```

```
2000 black 1
2001 red 2
2002 black 2
2003 white 4
2004 gray 2
2005 gray 2
2006 silver 2
2007 black 1
2008 white 5
2009 black 4
2010 white 5
2011 white 10
2012 white 27
2013 white 26
2014 white 30
```

```
2014 white 30
2015 white 55
2016 white 57
2017 white 99
2018 white 100
2019 white 271
2020 black 13
white       12
black        5
gray         2
red          1
silver       1
```

yes, although white was the dominant color in 12 years, the dominance wasn't complete.

What's the propability that a FORD car has a price more than the avarage price for all cars?

```
print(round(df[(df['brand']=='ford') &
                (df['price'] > df['price'].mean())].shape[0] /
              df[df['brand']=='ford'].shape[0] * 100,2),'%')
```

```
56.68 %
```

Answer = 56.68 %

Which model has the lowest mileage?

```python
#Which model has the lowest mileage?
df[df['mileage'] == df['mileage'].min()]['model'].unique()
# Door, Chassis, and Truck
```

```
array(['door', 'chassis', 'truck'], dtype=object)
```

Answer:
Door, Chassis, and Truck.

Which is the best selling brand in Virginia?

```
df[df['state'] == 'virginia'].value_counts('brand')
```

```
brand
ford                52
gmc                 10
dodge                9
nissan               8
honda                7
harley-davidson      1
chevrolet            1
cadillac             1
buick                1
dtype: int64
```

Answer: Ford

When was Chebrolet the best selling car?

```
#When was ford the best selling car?
for year in df['year'].unique():
  if df[df['year'] == year].value_counts('brand').idxmax() == 'chevrolet':
    print (year)
```

```
print (year)

2008
2011
2010
2009
1973
2006
2007
2004
1995
```

Answer:
In 2008, 2011, 2010, 2009, 1973, 2006, 2007, 2004, and 1995.

Which state has the highest number of DODGEs?

```
#Which state has the highest number of DODGEs?
df[df['brand'] == 'dodge'].value_counts('state').idxmax()
# pennsylvania
```

```
'pennsylvania'
```

Answer:
Pennsylvania

Train a machine learning model and evaluate its accuracy.

I trained a machine learning model to predict the price of the cars based on specific attributes of the cars.

Cleaning Data by dropping unwanted features:

```
df.drop(['Unnamed: 0','vin','lot'],axis=1,inplace=True)
df.head()
```

	price	brand	model	year	title_status	mileage	color	state	country	condition
0	6300	toyota	cruiser	2008	clean vehicle	274117.0	black	new jersey	usa	10 days left
1	2899	ford	se	2011	clean vehicle	190552.0	silver	tennessee	usa	6 days left
2	5350	dodge	mpv	2018	clean vehicle	39590.0	silver	georgia	usa	2 days left
3	25000	ford	door	2014	clean vehicle	64146.0	blue	virginia	usa	22 hours left
4	27700	chevrolet	1500	2018	clean vehicle	6654.0	red	florida	usa	22 hours left

We deleted 'Unnamed: 0' , as it is a simple index column. 'vin' and 'lot' had so many NaN values and were not useful, as they contain unique values for each car without any trends or patterns.

Converting condition into numerical values in minutes

condition
10 days left
6 days left
2 days left
22 hours left
22 hours left

```
df['value']= df['condition'] .str.split(' ').str[0]
df['days']= df['condition'] .str.split(' ').str[1]

def days_to_min_converter(time):
    return int(time)*1440

def hours_to_min_converter(time):
    return int(time)*60


temp_data=pd.concat([df[df['days']=='days']['value'].apply(days_to_min_converter),
           df[df['days']=='hours']['value'].apply(hours_to_min_converter),
           df[df['days']=='minutes']['value'].astype(int)]).rename('Minutes_Left',inplace=
True)



df=pd.concat([df,temp_data],axis=1)
df['Minutes_Left'].fillna(-200,inplace=True)

df.drop(['condition','value','days'],axis=1,inplace=True)
```

Minutes_Left
14400.0
8640.0
2880.0
1320.0
1320.0

Encoding catergorical features before training

```
categorical_features=[feature for feature in df.columns if df[feature].dtype=='O']

numerical_features=[feature for feature in df.columns if df[feature].dtype!='O']

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

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=0)

train_set=pd.concat([X_train,y_train],axis=1)
test_set=pd.concat([X_test,y_test],axis=1)
```

```
for feature in categorical_features:
    feature_labels=train_set.groupby(feature)['price'].mean().sort_values().index
    feature_labels={k:i for i,k in enumerate(feature_labels,0)}
    train_set[feature]=train_set[feature].map(feature_labels)
    test_set[feature]=test_set[feature].map(feature_labels)

test_set.dropna(inplace=True)

scaler=StandardScaler()

scaled_X_train=pd.DataFrame(scaler.fit_transform(train_set.drop('price',axis=1)), columns=X_train.columns)
scaled_X_train.index=train_set.index
scaled_X_test=pd.DataFrame(scaler.transform(test_set.drop('price',axis=1)), columns=X_test.columns)
scaled_X_test.index=test_set.index
scaled_train=pd.concat([scaled_X_train,train_set['price']],axis=1)
scaled_test=pd.concat([scaled_X_test,test_set['price']],axis=1)
X_train=scaled_train.drop('price',axis=1)
y_train=scaled_train['price']
X_test=scaled_test.drop('price',axis=1)
y_test=scaled_test['price']
```

Train the Models

```
def try_model(model):
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)
    pd.DataFrame(y_pred)
    return 'Model Testing Accurancy: ', r2_score(y_test, y_pred)
```

Accuracy Test

```
def try_model(model):
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)
    pd.DataFrame(y_pred)
    return 'Model Testing Accurancy: ', r2_score(y_test, y_pred)
```

```
neigh = KNeighborsRegressor(n_neighbors=6)
try_model(neigh)
```

```
('Model Testing Accurancy: ', 0.6945991238808297)
```

```
forest = RandomForestRegressor(max_depth=50, random_state=1)
try_model(forest)
```

```
('Model Testing Accurancy: ', 0.7542184266653602)
```

```
XGB = XGBRegressor(n_estimators=500, max_depth=20, eta=0.1, subsample=0.7, colsample_bytree=0.8)
try_model(XGB)
```

```
[22:03:32] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
('Model Testing Accurancy: ', 0.7631954182832582)('Model Testing Accurancy: ', 0.7631954182832582)
```

Highest Accuracy Possible; Compare with others at Kaggle Code Board for this dataset

Before you go!

Check our notebook on Kaggle and compare it with others:

[US Cars EDA & Price Prediction [Highest Accuracy]]

Check our full work notebook on Colab to see full code and documentation:

US_Cars_Notebook.ipynb

Work Was Done By ..

MOHAMED A. MOSTAFA

Computer Science Student @ Davidson College

Thank you!

Feel free to approach us if you have any questions.