Exam1

February 5, 2019

1 Exam 1

In this exam, you will work with a data set of 50000 used cars sold in the United States in 2018. The data set is available here:

`https://raw.githubusercontent.com/dlsun/data-science-book/master/data/usedcars.csv`

Answer the 8 questions below. The point values are clearly indicated next to each question. There are 30 possible points.

Some of the questions are deliberately vague. If you are not sure whether your answer is acceptable, make sure you document your thought process thoroughly in your explanation.

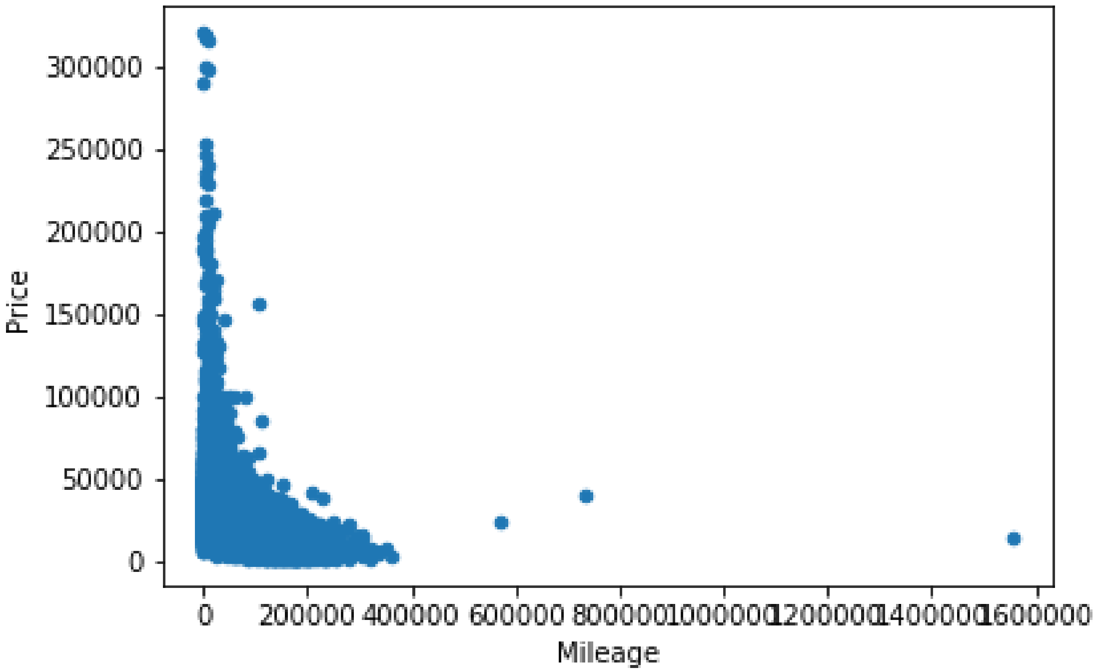
In [1]:
```
%matplotlib inline
import pandas as pd
import numpy as np

pd.options.display.max_rows = 30

cars = pd.read_csv("https://raw.githubusercontent.com/dlsun/data-science-book/master/da
```

2 Question 1 (3 points)

How is the mileage on a car related to its price? Make a visualization and report a summary statistic. What general trend do you notice?

In [2]:
```
cars.plot.scatter("Mileage", "Price")
cars["Price"].cov(cars["Mileage"])
```

Out[2]:
```
-236558058.88970679
```



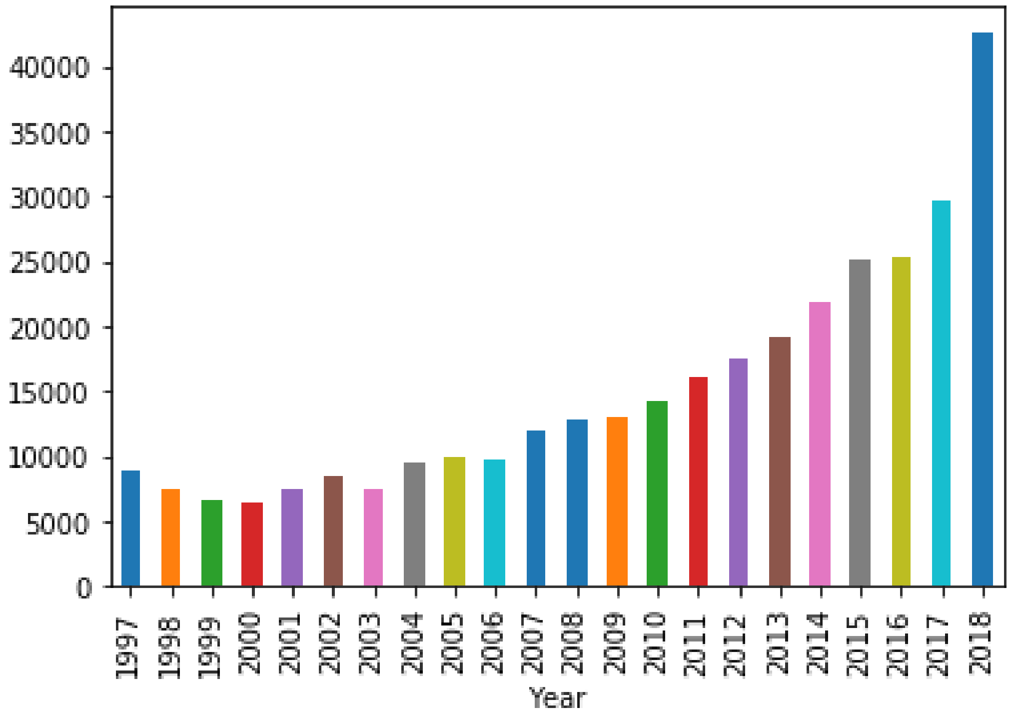
Overall, we see from the plot that cars with the least amount of mileage have the highest prices. We do see a few outliers where there are cars high mileage along with low prices as we would expect - the more miles, the lower the price. The covariance between Mileage and Price is negative meaning that as one variable increases, the other is likely to decrease.

3 Question 2 (3 points)

Make a visualization that shows the average price of a car by year. What general trend do you notice?

```
In [3]: cars.groupby(["Year"])["Price"].mean().plot.bar()
```

```
Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x7fea90733dd8>
```



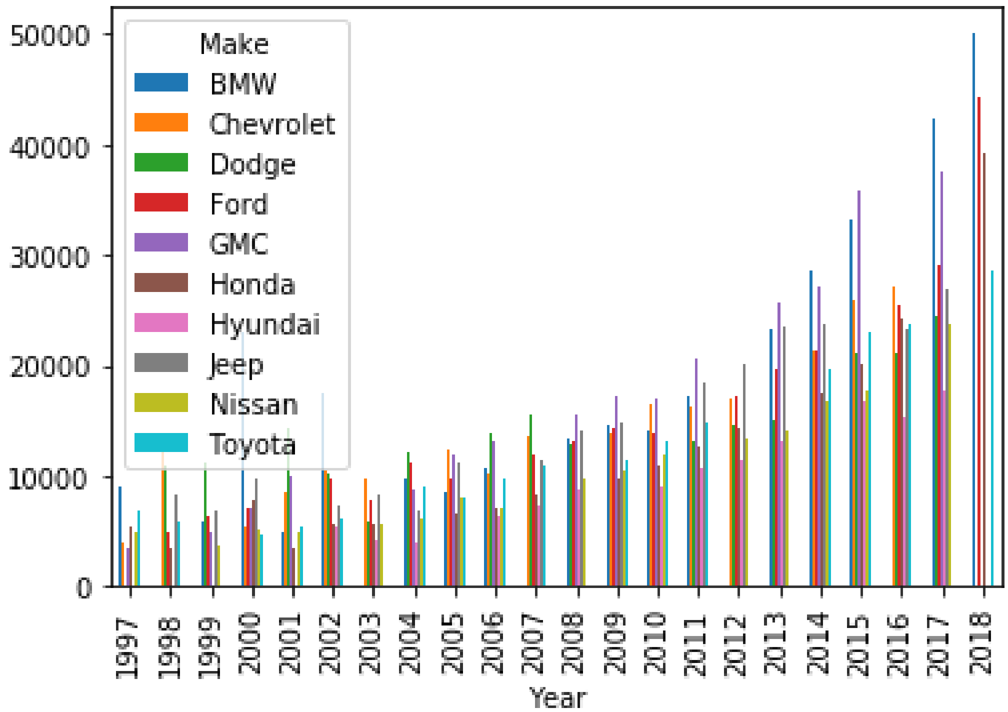
As the year increases, we see that the average prices of cars increases. The rise in prices from the distribution appears to be exponential as the average price of cars from 2018 is over 4x greater than the average price of cars from 1997.

4 Question 3 (4 points)

Restrict to top 10 makes (i.e., the 10 makes that appeared the most times) in this data set. Make a graphic that shows how the average price of each make of car changed by year. Your graphic should make it just as easy to compare the different makes as the different years. Explain what you see.

```
In [4]: cars.Make.value_counts().iloc[0:10]

        top10 = cars[(cars.Make == "Ford") | (cars.Make == "Chevrolet") |
                     (cars.Make == "Toyota") | (cars.Make == "Nissan") |
                    (cars.Make == "Honda") |  (cars.Make == "Jeep") |
                     (cars.Make == "Hyundai") | (cars.Make == "Dodge") |
                   (cars.Make == "BMW") | (cars.Make == "GMC")]

        top10.pivot_table(
            index="Year", columns=["Make"],
            values="Price", aggfunc=np.mean
        ).plot.bar()
```

Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x7fea90610ef0>



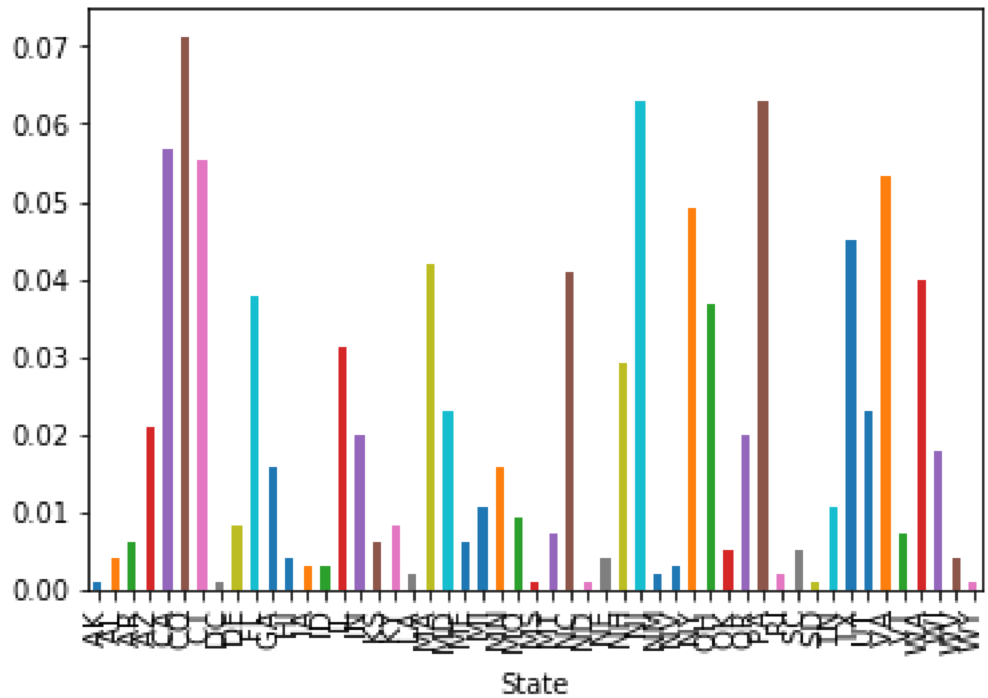
From the distribution, we see that as the time continues, the average price of cars overall continues to increase. We see trends in the data from year to year where one Make has a higher average price than other makes and then the next year, a different make has the highest average price - for example Dodge from 2007 to 2008. In more recent years, BMW has had the highest average price of cars, while Toyota has continually has had one of the lower average priced vehicles.

5 Question 4 (4 points)

I recently learned the stereotype that people from Colorado like to drive Subarus. Does this appear to be true? Calculate appropriate conditional distributions to assess this claim.

In [5]:
```
cars.State = cars.State.str.upper()

make_state_counts = pd.crosstab(cars.Make, cars.State)
make_state_counts

make_counts = make_state_counts.sum(axis=1)
make_counts
make_given_state =  make_state_counts.divide(make_counts, axis=0)
make_given_state.loc["Subaru"].plot.bar()
make_given_state.loc["Subaru"]
```

Out[5]:
```
State
 AK    0.001047
 AL    0.004188
 AR    0.006283
 AZ    0.020942
 CA    0.056545
 CO    0.071204
 CT    0.055497
 DC    0.001047
 DE    0.008377
 FL    0.037696
 GA    0.015707
 HI    0.004188
 IA    0.003141
 ID    0.003141
 IL    0.031414
         ...
 OK    0.005236
 OR    0.019895
 PA    0.062827
 RI    0.002094
 SC    0.005236
 SD    0.001047
 TN    0.010471
 TX    0.045026
 UT    0.023037
 VA    0.053403
 VT    0.007330
 WA    0.039791
 WI    0.017801
 WV    0.004188
 WY    0.001047
Name: Subaru, Length: 51, dtype: float64
```



According to the distribution, it does appear that given that a person is from Colorado, he or she is more likely to drive a Subaru. The brown bar on the left of the plot is Colorado and we see that the proportion of Subaru drivers is highest here than in any other state. However, states such as Connecticut, Pennsylvania, and Virgina are nearly just as high - all within 2%.

6 Question 5 (4 points)

Calculate the joint distribution between the year and the 10th digit of the VIN number. What do you notice? Can you explain why this is?

```
In [6]: cars.Vin.head()

Out[6]: 0    JTJJM7FX5A5009042
        1    WDDHF5GB9AA106034
        2    1FM5K8F82HGC23716
        3    1C4RJFBG9EC506630
        4    1FADP3K24FL209932
        Name: Vin, dtype: object

In [7]: cars["tenth_char"] = cars.Vin.str[9]

        pd.crosstab(cars.tenth_char, cars.Year,
                    normalize=True)
```

```
Out[7]: Year            1997    1998     1999     2000     2001     2002     2003  \
tenth_char
1            0.00000  0.0000  0.00000  0.00000  0.00296  0.00000  0.00000
2            0.00000  0.0000  0.00000  0.00000  0.00000  0.00404  0.00000
3            0.00000  0.0000  0.00000  0.00000  0.00000  0.00000  0.00662
4            0.00000  0.0000  0.00000  0.00000  0.00000  0.00000  0.00000
5            0.00000  0.0000  0.00000  0.00000  0.00000  0.00000  0.00000
6            0.00000  0.0000  0.00000  0.00000  0.00000  0.00000  0.00000
7            0.00000  0.0000  0.00000  0.00000  0.00000  0.00000  0.00000
8            0.00000  0.0000  0.00000  0.00000  0.00000  0.00000  0.00000
9            0.00000  0.0000  0.00000  0.00000  0.00000  0.00000  0.00000
A            0.00000  0.0000  0.00000  0.00000  0.00000  0.00000  0.00000
B            0.00000  0.0000  0.00000  0.00000  0.00000  0.00000  0.00000
C            0.00000  0.0000  0.00000  0.00000  0.00000  0.00000  0.00000
D            0.00000  0.0000  0.00000  0.00000  0.00000  0.00000  0.00000
E            0.00000  0.0000  0.00000  0.00000  0.00000  0.00000  0.00000
F            0.00000  0.0000  0.00000  0.00000  0.00000  0.00000  0.00000
G            0.00000  0.0000  0.00000  0.00000  0.00000  0.00000  0.00000
H            0.00000  0.0000  0.00000  0.00000  0.00000  0.00000  0.00000
J            0.00000  0.0000  0.00000  0.00000  0.00000  0.00000  0.00000
V            0.00086  0.0000  0.00000  0.00000  0.00000  0.00000  0.00000
W            0.00000  0.0011  0.00000  0.00000  0.00000  0.00000  0.00000
X            0.00000  0.0000  0.00182  0.00000  0.00000  0.00000  0.00000
Y            0.00000  0.0000  0.00000  0.00232  0.00000  0.00000  0.00000

Year            2004     2005     2006   ...      2009     2010     2011  \
tenth_char                               ...
1            0.00000  0.00000  0.00000   ...   0.00000  0.00000  0.00000
2            0.00000  0.00000  0.00000   ...   0.00000  0.00000  0.00000
3            0.00000  0.00000  0.00000   ...   0.00000  0.00000  0.00000
4            0.00974  0.00000  0.00000   ...   0.00000  0.00000  0.00000
5            0.00000  0.01246  0.00000   ...   0.00000  0.00000  0.00000
6            0.00000  0.00000  0.01702   ...   0.00000  0.00000  0.00000
7            0.00000  0.00000  0.00000   ...   0.00000  0.00000  0.00000
8            0.00000  0.00000  0.00000   ...   0.00000  0.00000  0.00000
9            0.00000  0.00000  0.00000   ...   0.02336  0.00000  0.00000
A            0.00000  0.00000  0.00000   ...   0.00000  0.03114  0.00000
B            0.00000  0.00000  0.00000   ...   0.00000  0.00000  0.04634
C            0.00000  0.00000  0.00000   ...   0.00000  0.00000  0.00000
D            0.00000  0.00000  0.00000   ...   0.00000  0.00000  0.00000
E            0.00000  0.00000  0.00000   ...   0.00000  0.00000  0.00000
F            0.00000  0.00000  0.00000   ...   0.00000  0.00000  0.00000
G            0.00000  0.00000  0.00000   ...   0.00000  0.00000  0.00000
H            0.00000  0.00000  0.00000   ...   0.00000  0.00000  0.00000
J            0.00000  0.00000  0.00000   ...   0.00000  0.00000  0.00000
V            0.00000  0.00000  0.00000   ...   0.00000  0.00000  0.00000
W            0.00000  0.00000  0.00000   ...   0.00000  0.00000  0.00000
X            0.00000  0.00000  0.00000   ...   0.00000  0.00000  0.00000
```

```
Y           0.00000  0.00000  0.00000    ...     0.00000  0.00000  0.00000

Year           2012     2013     2014     2015     2016     2017     2018
tenth_char
1           0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
2           0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
3           0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
4           0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
5           0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
6           0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
7           0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
8           0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
9           0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
A           0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
B           0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
C           0.05728  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
D           0.00000  0.08996  0.00002  0.00000  0.00000  0.00000  0.00000
E           0.00000  0.00000  0.18958  0.00000  0.00000  0.00000  0.00000
F           0.00000  0.00000  0.00000  0.18328  0.00000  0.00000  0.00000
G           0.00000  0.00000  0.00000  0.00000  0.15914  0.00000  0.00000
H           0.00000  0.00000  0.00000  0.00000  0.00000  0.10678  0.00000
J           0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00116
V           0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
W           0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
X           0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
Y           0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000

[22 rows x 22 columns]
```

The joint distribution of the 10th character in the VIN and the year has many 0's as entries. We would expect to see some 0's because there are technically there should be 36 different characters to choose from (26 letters and 10 digits). However, the high number of 0's from the 22 characters in the data set indicates that VIN characters are not being equally distributed to all cars. As a matter of fact, there is a particular trend in the 10th character of the VIN for cars in this data set. For each year in the data, the 10th character of the VIN is just moving down one row, and across one column, following a diagonal pattern. Once the pattern reaches the bottom row, the following year, the pattern restarts at the 1st row. Therefore, the 10th character of VINs from this data of cars is not selected at random.

7 Question 6 (4 points)

Which states tend to use their cars the most? Calculate the mileage per year of each car in the data set. (*Reminder:* This data set was collected in 2018.) Then, make a visualization that shows the average mileage per year by state, sorted by state. What do you notice?

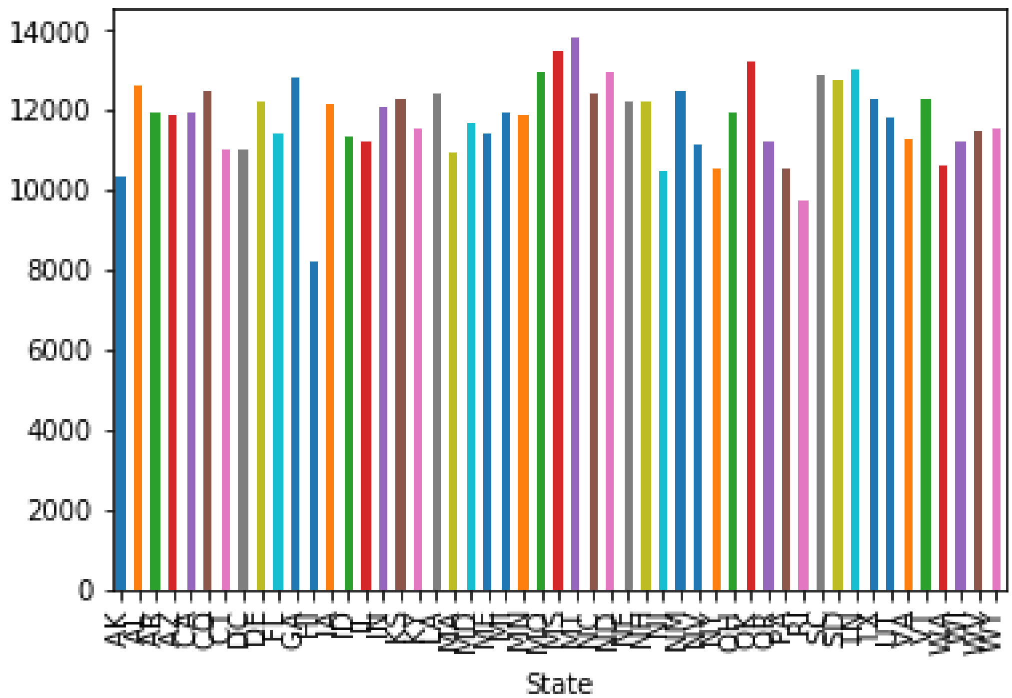
```
In [8]: cars_not_2018 = cars[cars.Year != 2018]
        cars_not_2018["MilesPerYear"] = cars_not_2018.Mileage/(2018 - cars_not_2018.Year)
```

```
cars_2018 = cars[cars.Year == 2018]
cars_2018["MilesPerYear"] = cars_2018.Mileage

all_cars = cars_not_2018.append(cars_2018)

all_cars.head()

all_cars.groupby(["State"])["MilesPerYear"].mean().plot.bar()
(all_cars.groupby(["State"])["MilesPerYear"].mean().idxmax(),
all_cars.groupby(["State"])["MilesPerYear"].mean().max())
#all_cars.groupby(["State"])["MilesPerYear"].mean().idxmin()

all_cars.groupby(["State"])["MilesPerYear"].mean().sort_values()
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html

/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  """
```

```
Out[8]: State
         HI     8195.416301
         RI     9720.249827
         AK    10365.474103
         NJ    10513.521222
         NY    10554.097645
         PA    10572.333259
         WA    10613.537904
         MA    10928.066531
         CT    11029.463851
         DC    11037.185119
         NV    11135.664437
         WI    11188.688604
         IL    11191.288817
         OR    11195.453171
         VA    11257.515745
                   ...
         TX    12297.402497
         NC    12385.922396
         LA    12426.400184
```

```
NM     12486.916955
CO     12512.975762
AL     12651.660493
SD     12769.054951
GA     12825.340939
SC     12878.410567
MO     12935.415172
ND     12949.837128
TN     13013.985716
OK     13233.242244
MS     13482.717784
MT     13832.694185
Name: MilesPerYear, Length: 51, dtype: float64
```



From the bar graph, it appears that most drivers on average use their cars roughly the same amount with exception to drivers from Hawaii. The state with the highest miles per year average is Montana with nearly 14000 miles driven per year on average. However, drivers from Mississippi, Oklahoma, and Tennesse are not far behind - all within an average of 1000 miles.

8 Question 7 (4 points)

Suppose you are moving from San Luis Obispo to Houston, TX. You put your 2005 Porsche on sale (observation 8111 in the `DataFrame`) and would like to find a similar used car in Houston. Which car on sale in Houston is most similar to your current car? Is this car a Porsche?

In [9]:
```
cars.iloc[8111]
```

Out[9]:
```
Price                     34040
Year                       2005
Mileage                   52851
City            San Luis Obispo
State                        CA
Vin           WP0AA29975S715237
Make                    Porsche
Model                    9112dr
tenth_char                    5
Name: 8111, dtype: object
```

In [10]:
```
houston_only = cars[cars.City == "Houston"]
houston_only = houston_only.append(cars.iloc[8111])
houston_only.loc[8111]


houston_num = pd.get_dummies(
    houston_only.drop(["Vin", "tenth_char"], axis=1))


dist_norm = np.sqrt(((houston_num - houston_num.loc[8111]) **
                     2).sum(axis=1)).sort_values()

print(dist_norm.index[:5])
#houston_num.loc[dist_norm.index[:5]]

cars.loc[dist_norm.index[:10]]
```

```
Int64Index([8111, 41054, 16806, 15231, 25592], dtype='int64')
```

Out[10]:
```
       Price  Year  Mileage             City State                Vin  \
8111   34040  2005    52851  San Luis Obispo    CA  WP0AA29975S715237
41054  33900  2012    52650          Houston    TX  1GNSKBE0XCR161261
16806  30995  2015    53263          Houston    TX  3GCPCREC8FG203526
15231  33990  2016    56711          Houston    TX  5TFEM5F13GX095792
25592  35998  2015    49442          Houston    TX  1C6RR7LTXFS788155
29976  29991  2014    51342          Houston    TX  1GKKRTKD5EJ296851
49392  30000  2014    49634          Houston    TX  5N1AL0MN7EC543705
20667  29958  2010    56439          Houston    TX  WDDNG8GB8AA358407
38805  27888  2015    51250          Houston    TX  2G1FK3DJ0F9175932
35989  27950  2014    55603          Houston    TX  1C6RR6MT8ES102220

            Make          Model tenth_char
8111     Porsche         9112dr          5
41054  Chevrolet       Tahoe4WD          C
16806  Chevrolet      Silverado          F
```

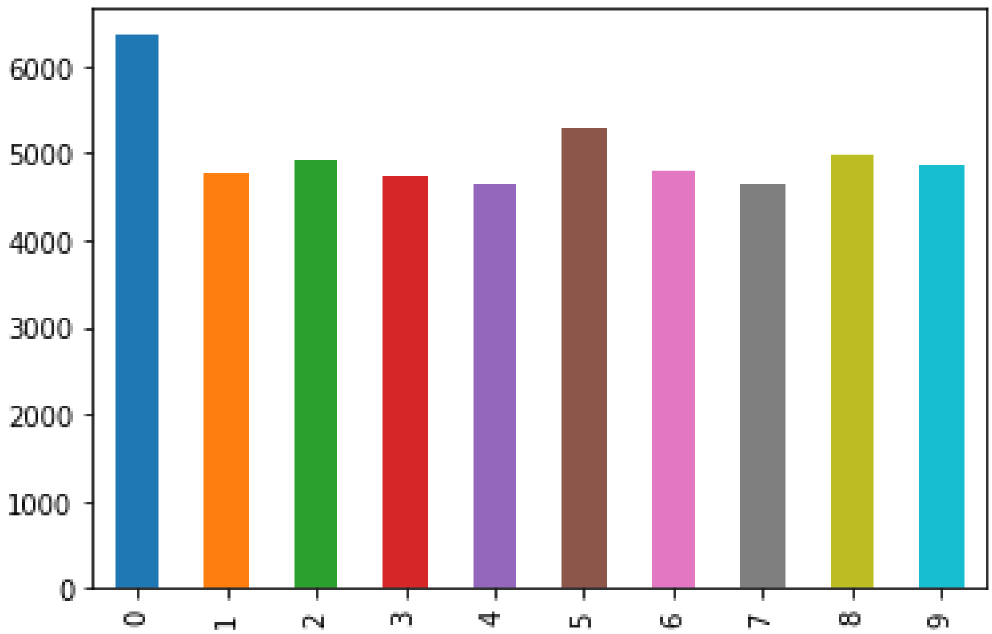
```
15231          Toyota             Tundra          G
25592             Ram            1500Lone         F
29976             GMC            AcadiaFWD        E
49392        INFINITI             QX602WD         E
20667   Mercedes-Benz           S-Class4dr        A
38805       Chevrolet   CamaroConvertible         F
35989             Ram           1500Sport,        E
```

The most similar car on sale in Houston is a Chevrolet Tahoe with slightly less miles and it is also selling for a lower price than what I am selling my Porshe for. However, just based on pure looks and aesthetics of the vehicle, I would be going from a sports car to a Truck which is what I might not be looking for. As a matter of fact, many of the top cars that are "closest" to mine are trucks or are SUVs. When I reach the 7th and 8th closest cars, I come across more "sporty" cars that also have similar mileage, price, and year. These vehicles, the Mercedes-Benz S-Class and Chevrolet Camaro, are both cheaper than my car, have less miles than my car, and are newer than my car, so it might actually be a better fit.

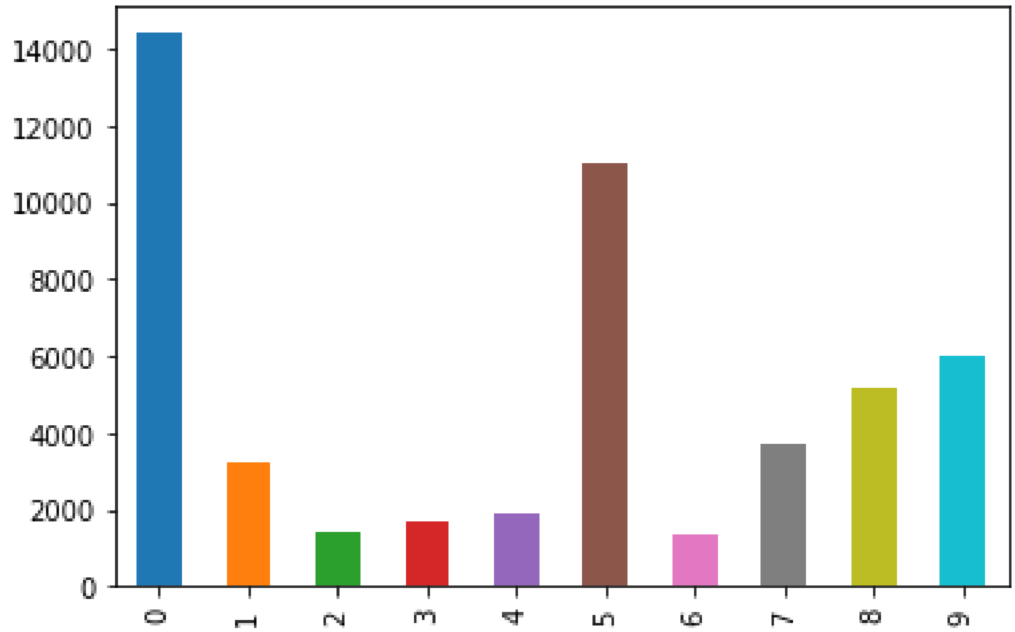
9 Question 8 (4 points)

Make a graphic that shows how the last digits of prices and the last digits of mileages are distributed. Do they appear to be uniformly distributed over the digits 0-9? How do the two distributions compare to each other?

```
In [11]: last_digit_miles = cars.Mileage.astype(str).str[-1]

In [12]: last_digit_prices = cars.Price.astype(str).str[-1]

In [13]: last_digit_miles.value_counts().sort_index().plot.bar()

         last_digit_miles.value_counts()/len(last_digit_miles)

Out[13]: 0    0.12706
         5    0.10592
         8    0.09970
         2    0.09814
         9    0.09698
         6    0.09628
         1    0.09524
         3    0.09466
         7    0.09308
         4    0.09294
         Name: Mileage, dtype: float64
```




```
In [14]: last_digit_prices.value_counts().sort_index().plot.bar()

         last_digit_prices.value_counts()/len(last_digit_prices)

Out[14]: 0    0.28808
         5    0.22108
         9    0.12082
         8    0.10342
         7    0.07394
         1    0.06410
         4    0.03860
         3    0.03396
         2    0.02860
         6    0.02740
         Name: Price, dtype: float64
```



The distribution of the last digit in miles appears to be almost uniform. The most common last digit is 0, but is only 3% greater than 4 (the least common last digit in miles).

The distribution of the last digit in prices appears to be not uniform. The most common last digits are 0, 5, 9 which makes sense because often times in car lots, the sales prices are rounded to the nearest tens place or dollar. The digits 0, 5, 9 account for nearly 65% of the distribution of the last digit in prices.

Therefore, these 2 graphs are not the same because the mileage graph is nearly uniform while the price graph is not uniform.

10 Submission Instructions

Once you are finished, follow these steps:

1. Restart the kernel and re-run this notebook from beginning to end by going to `Kernel > Restart Kernel and Run All Cells`.
2. If this process stops halfway through, that means there was an error. Correct the error and repeat Step 1 until the notebook runs from beginning to end.
3. Double check that there is a number next to each code cell and that these numbers are in order.

Then, submit your exam as follows:

1. Go to `File > Export Notebook As > PDF`.
2. Double check that the entire notebook, from beginning to end, is in this PDF file. (If the notebook is cut off, try first exporting the notebook to HTML and printing to PDF.)
3. Upload the PDF to PolyLearn.