Used Car Price Analysis

October 2, 2024

1 Final project

[1]:
```
from os.path import basename, exists


def download(url):
    filename = basename(url)
    if not exists(filename):
        from urllib.request import urlretrieve

        local, _ = urlretrieve(url, filename)
        print("Downloaded " + local)


download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/
 ↪thinkstats2.py")
download("https://github.com/AllenDowney/ThinkStats2/raw/master/code/thinkplot.
 ↪py")
```

[2]:
```
import pandas as pd
import numpy as np
import thinkplot
import thinkstats2
from scipy import stats
import matplotlib.pyplot as plt
from matplotlib.pyplot import hist, xticks, show
import statsmodels.formula.api as smf
```

[3]:
```
df = pd.read_csv("/Users/yuhang/Desktop/DSC530/FINAL/USA_cars_datasets.csv")
```

[4]:
```
df.head()
```

[4]:

	Unnamed: 0	price	brand	model	year	title_status	mileage	\
0	0	6300	toyota	cruiser	2008	clean vehicle	274117.0	
1	1	2899	ford	se	2011	clean vehicle	190552.0	
2	2	5350	dodge	mpv	2018	clean vehicle	39590.0	
3	3	25000	ford	door	2014	clean vehicle	64146.0	

```
4           4  27700  chevrolet     1500  2018  clean vehicle    6654.0

    color                 vin        lot       state country      condition
0   black    jtezu11f88k007763  159348797  new jersey     usa   10 days left
1  silver    2fmdk3gc4bbb02217  166951262   tennessee     usa    6 days left
2  silver    3c4pdcgg5jt346413  167655728     georgia     usa    2 days left
3    blue    1ftfw1et4efc23745  167753855    virginia     usa  22 hours left
4     red    3gcpcrec2jg473991  167763266     florida     usa  22 hours left
```

I'm going to use 'brand', 'year', 'mileage', 'color' and 'state' to make analysis of the price. The variable 'brand' is the car brand, 'mileage' is the miles traveled by vehicle, 'year' is the the vehicle registration year, 'color' is the color of the vehicle, while 'state' is the state in which the car is being available for purchase.

First, I neeed to do some transformation of the variables. Since there are 28 car brands, I'd encode them to numbers.

1.0.1 data clean

[5]:
```
# df["brand"].tolist()
```

[6]:
```
# encode brand to number
```

[7]:
```
brand_num = range(1,1+len(df["brand"].unique()))
brand_to_num = dict(zip(df["brand"].unique(),brand_num))


df["brand_to_num"] = df["brand"]
df['brand_to_num'] = df['brand_to_num'].map(brand_to_num)
```

[8]:
```
print(brand_to_num)
```

```
{'toyota': 1, 'ford': 2, 'dodge': 3, 'chevrolet': 4, 'gmc': 5, 'chrysler': 6,
'kia': 7, 'buick': 8, 'infiniti': 9, 'mercedes-benz': 10, 'jeep': 11, 'bmw': 12,
'cadillac': 13, 'hyundai': 14, 'mazda': 15, 'honda': 16, 'heartland': 17,
'jaguar': 18, 'acura': 19, 'harley-davidson': 20, 'audi': 21, 'lincoln': 22,
'lexus': 23, 'nissan': 24, 'land': 25, 'maserati': 26, 'peterbilt': 27, 'ram':
28}
```

[9]:
```
# encode 'state' by zone
```

[10]:
```
us_state_to_abbrev = {
    "alabama": "AL","alaska": "AK","arizona": "AZ","arkansas":␣
 ↪"AR","california": "CA","colorado": "CO","connecticut": "CT",
    "delaware": "DE","florida": "FL","georgia": "GA","hawaii": "HI","idaho":␣
 ↪"ID","illinois": "IL","indiana": "IN",
    "iowa": "IA","kansas": "KS","kentucky": "KY","louisiana": "LA","maine":␣
 ↪"ME","maryland": "MD","massachusetts": "MA",
```

```
    "michigan": "MI","minnesota": "MN","mississippi": "MS","missouri":␣
 ↪"MO","montana": "MT","nebraska": "NE","nevada": "NV",
    "new hampshire": "NH","new jersey": "NJ","new mexico": "NM","new york":␣
 ↪"NY","north carolina": "NC","north dakota": "ND",
    "ohio": "OH","oklahoma": "OK","oregon": "OR","pennsylvania": "PA","rhode␣
 ↪island": "RI","south carolina": "SC",
    "south dakota": "SD","tennessee": "TN","texas": "TX","utah": "UT","vermont":
 ↪ "VT","virginia": "VA","washington": "WA",
    "west virginia": "WV","wisconsin": "WI","wyoming": "WY","district of␣
 ↪Columbia": "DC","american Samoa": "AS","guam": "GU",
    "northern Mariana Islands": "MP","puerto Rico": "PR","United States Minor␣
 ↪Outlying Islands": "UM","U.S. Virgin Islands": "VI","ontario":"ON"
}
```

[11]:
```
df["state_abbrev"] = df["state"]
df['state_abbrev'] = df['state_abbrev'].map(us_state_to_abbrev)
```

[12]:
```
state2timezone = {'ON': 'US/Eastern','AK': 'US/Alaska', 'AL': 'US/Central',␣
 ↪'AR': 'US/Central', 'AS': 'US/Samoa', 'AZ': 'US/Mountain', 'CA': 'US/
 ↪Pacific', 'CO': 'US/Mountain', 'CT': 'US/Eastern', 'DC': 'US/Eastern', 'DE':␣
 ↪'US/Eastern', 'FL': 'US/Eastern', 'GA': 'US/Eastern', 'GU': 'Pacific/Guam',␣
 ↪'HI': 'US/Hawaii', 'IA': 'US/Central', 'ID': 'US/Mountain', 'IL': 'US/
 ↪Central', 'IN': 'US/Eastern', 'KS': 'US/Central', 'KY': 'US/Eastern', 'LA':␣
 ↪'US/Central', 'MA': 'US/Eastern', 'MD': 'US/Eastern', 'ME': 'US/Eastern',␣
 ↪'MI': 'US/Eastern', 'MN': 'US/Central', 'MO': 'US/Central', 'MP': 'Pacific/
 ↪Guam', 'MS': 'US/Central', 'MT': 'US/Mountain', 'NC': 'US/Eastern', 'ND':␣
 ↪'US/Central', 'NE': 'US/Central', 'NH': 'US/Eastern', 'NJ': 'US/Eastern',␣
 ↪'NM': 'US/Mountain', 'NV': 'US/Pacific', 'NY': 'US/Eastern', 'OH': 'US/
 ↪Eastern', 'OK': 'US/Central', 'OR': 'US/Pacific', 'PA': 'US/Eastern', 'PR':␣
 ↪'America/Puerto_Rico', 'RI': 'US/Eastern', 'SC': 'US/Eastern', 'SD': 'US/
 ↪Central', 'TN': 'US/Central', 'TX': 'US/Central', 'UT': 'US/Mountain', 'VA':␣
 ↪'US/Eastern', 'VI': 'America/Virgin', 'VT': 'US/Eastern', 'WA': 'US/
 ↪Pacific', 'WI': 'US/Central', 'WV': 'US/Eastern', 'WY': 'US/Mountain', '' :␣
 ↪'US/Pacific', '--': 'US/Pacific' }
```

[13]:
```
df["zone"] = df["state_abbrev"]
df['zone'] = df['zone'].map(state2timezone)
```

[14]:
```
zone_num = range(1,1+len(df["zone"].unique()))
zone_to_num = dict(zip(df["zone"].unique(),zone_num))
df["zone_to_num"] = df["zone"]
df['zone_to_num'] = df['zone_to_num'].map(zone_to_num)
```

[15]:
```
# sort 'color'
```

[16]:
```
color_category = {
```

```
        'black':'black', 'silver':'silver', 'blue':'blue', 'red':'red', 'white':
 ↪'white', 'gray':'gray', 'orange':'orange',
        'brown':'brown', 'no_color':'other', 'gold':'gold', 'charcoal':'black',␣
 ↪'turquoise':'blue', 'beige':'brown',
        'green':'green', 'dark blue':'blue', 'maroon':'red', 'phantom black':
 ↪'black', 'yellow':'yellow','color:':'other',
        'light blue':'blue', 'toreador red':'red', 'bright white clearcoat':
 ↪'white','billet silver metallic clearcoat':'silver',
        'black clearcoat':'black','jazz blue pearlcoat':'blue', 'purple':
 ↪'other','ruby red metallic tinted clearcoat':'red',
        'triple yellow tri-coat':'yellow','competition orange':'orange',␣
 ↪'off-white':'white', 'shadow black':'black',
        'magnetic metallic':'other', 'ingot silver metallic':'silver', 'ruby␣
 ↪red':'red',
        'royal crimson metallic tinted clearcoat':'other', 'kona blue metallic':
 ↪'blue','oxford white':'white',
         'lightning blue':'blue', 'ingot silver':'other','white platinum␣
 ↪tri-coat metallic':'white', 'guard':'other',
        'tuxedo black metallic':'black', 'tan':'other', 'burgundy':'other',␣
 ↪'super black':'black',
        'cayenne red':'red', 'morningsky blue':'blue', 'pearl white':'white',␣
 ↪'glacier white':'white'
}
```

[17]:
```
df["color_Cate"] = df["color"]
df['color_Cate'] = df['color_Cate'].map(color_category)
```

[19]:
```
# Encode 'color' to number
```

[20]:
```
color_num = range(1,1+len(df["color_Cate"].unique()))
color_to_num = dict(zip(df["color_Cate"].unique(),color_num))
df["color_to_num"] = df["color_Cate"]
df['color_to_num'] = df['color_to_num'].map(color_to_num)
```

[21]:
```
print(color_to_num)
```

```
{'black': 1, 'silver': 2, 'blue': 3, 'red': 4, 'white': 5, 'gray': 6, 'orange':
7, 'brown': 8, 'other': 9, 'gold': 10, 'green': 11, 'yellow': 12}
```

[22]:
```
# df.to_csv("df.csv")
```

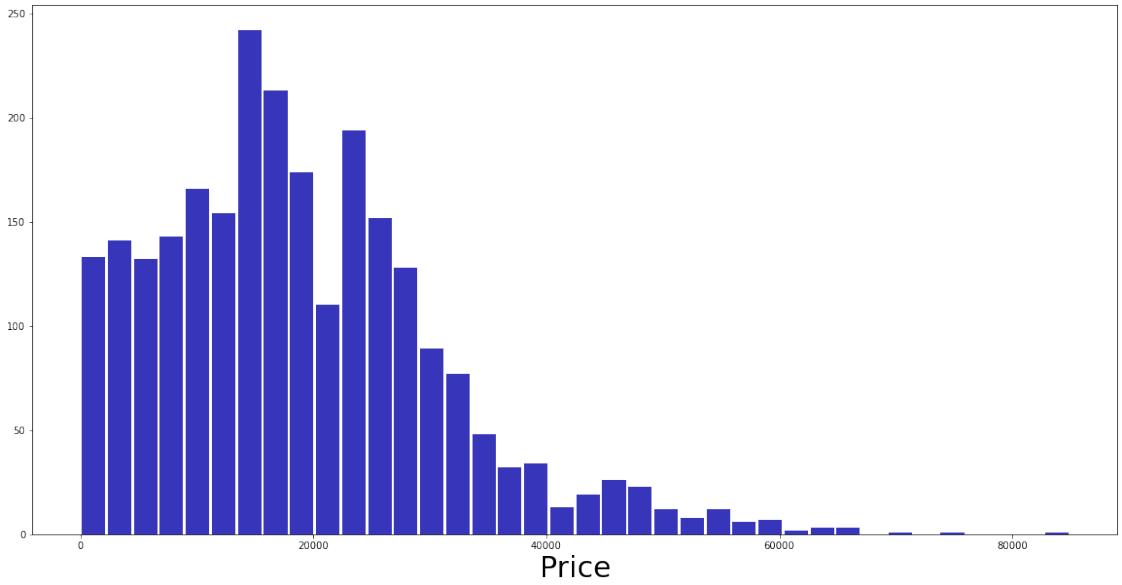
1.0.2 Plot histogram and descriptive characteristics

[23]:
```
# histogram of 'price'
```

[24]:
```
f = plt.figure()
f.set_figwidth(20)
```

```
f.set_figheight(10)
price_bins= np.arange(0, 100000, 20000)
n, bins, patches = plt.hist(x=df['price'],bins='auto', color='#0504aa',
                            alpha=0.8, rwidth=0.9)
plt.rc('xtick', labelsize=25)
plt.rc('ytick', labelsize = 35)
plt.xlabel('Price', fontsize=30)
```

[24]: Text(0.5, 0, 'Price')



[25]:
```
df.price.mean()
```

[25]: 18767.671468587436

[26]:
```
df.price.var()
```

[26]: 146799756.489702

[27]:
```
df.price.std()
```

[27]: 12116.094935650759

[28]:
```
df.mileage.mean()
```

[28]: 52298.685474189675

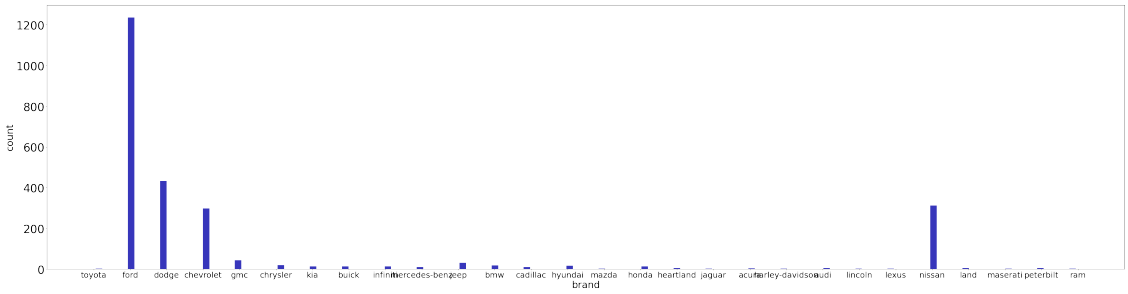
[29]:
```
stats.mode(df['price'])
```

[29]:
```
ModeResult(mode=array([0]), count=array([43]))
```

[30]:
```
# histogram of 'brand'
```

[31]:
```
f = plt.figure()
f.set_figwidth(60)
f.set_figheight(15)
n, bins, patches = plt.hist(x=df['brand'],bins='auto', color='#0504aa',
                            alpha=0.8, rwidth=0.6)
plt.rc('xtick', labelsize=25)
plt.rc('ytick', labelsize=25)
plt.xlabel('brand', fontsize=30)
plt.ylabel('count', fontsize=30)
```

[31]:
```
Text(0, 0.5, 'count')
```



[32]:
```
df.brand_to_num.mean()
```

[32]:
```
5.952781112444978
```

[33]:
```
df.brand_to_num.median()
```

[33]:
```
3.0
```

[34]:
```
stats.mode(df['brand'])
```

[34]:
```
ModeResult(mode=array(['ford'], dtype=object), count=array([1235]))
```

[35]:
```
df.brand_to_num.var()
```

[35]:
```
55.2547750885782
```

[36]:
```
df.brand_to_num.std()
```

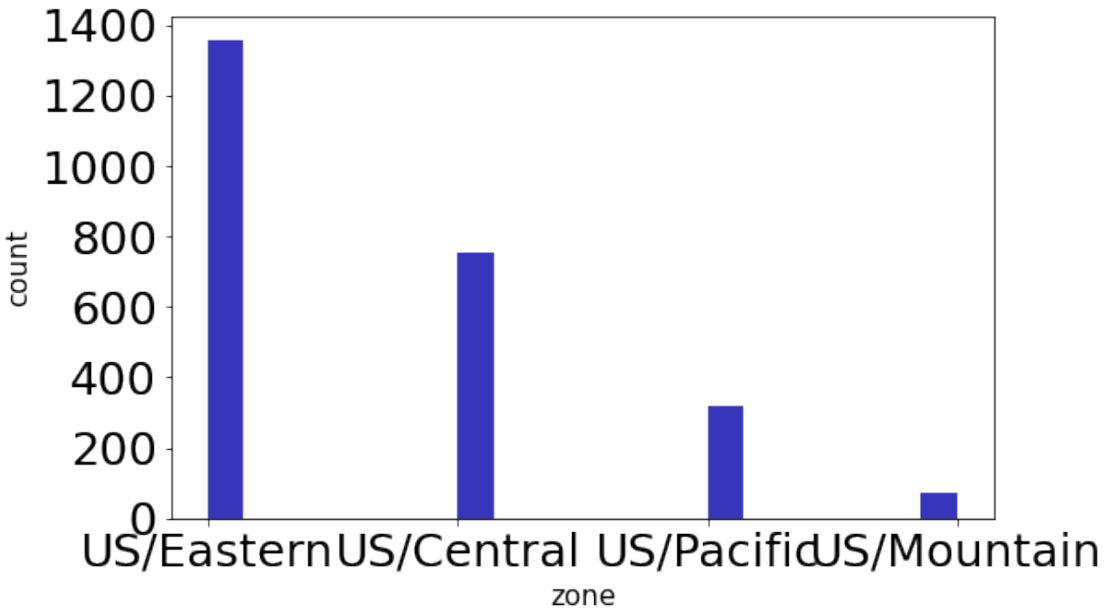
[36]:
```
7.433355573936861
```

[37]:
```
# histogram of 'zone'
```

[38]:
```
f = plt.figure()
f.set_figwidth(8)
f.set_figheight(5)

n, bins, patches = plt.hist(x=df['zone'],bins='auto', color='#0504aa',
                            alpha=0.8, rwidth=10)
plt.rc('xtick', labelsize=15)
plt.rc('ytick', labelsize=10)
plt.xlabel('zone', fontsize=15)
plt.ylabel('count', fontsize=15)
```

[38]:
```
Text(0, 0.5, 'count')
```



[39]:
```
stats.mode(df['state'])
```

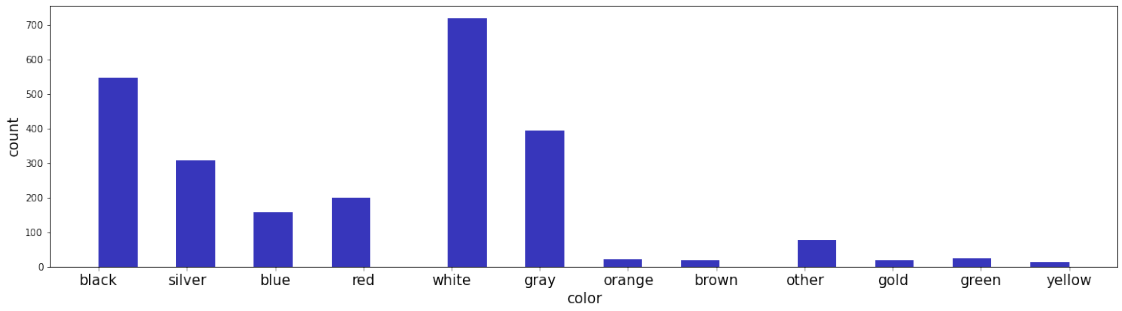
[39]:
```
ModeResult(mode=array(['pennsylvania'], dtype=object), count=array([299]))
```

[40]:
```
# histogram of 'color'
```

[41]:
```
f = plt.figure()
f.set_figwidth(20)
f.set_figheight(5)
n, bins, patches = plt.hist(x=df['color_Cate'],bins='auto', color='#0504aa',
                            alpha=0.8, rwidth=10)
plt.rc('xtick', labelsize=15)
plt.rc('ytick', labelsize=10)
plt.xlabel('color', fontsize=15)
```

```
plt.ylabel('count', fontsize=15)
```

[41]: Text(0, 0.5, 'count')



[42]:
```
df.color_to_num.mean()
```

[42]: 4.002801120448179

[43]:
```
df.color_to_num.median()
```

[43]: 5.0

[44]:
```
stats.mode(df['color'])
```

[44]: ModeResult(mode=array(['white'], dtype=object), count=array([707]))

[45]:
```
df.color_to_num.var()
```

[45]: 5.4823780593101255

[46]:
```
df.color_to_num.std()
```

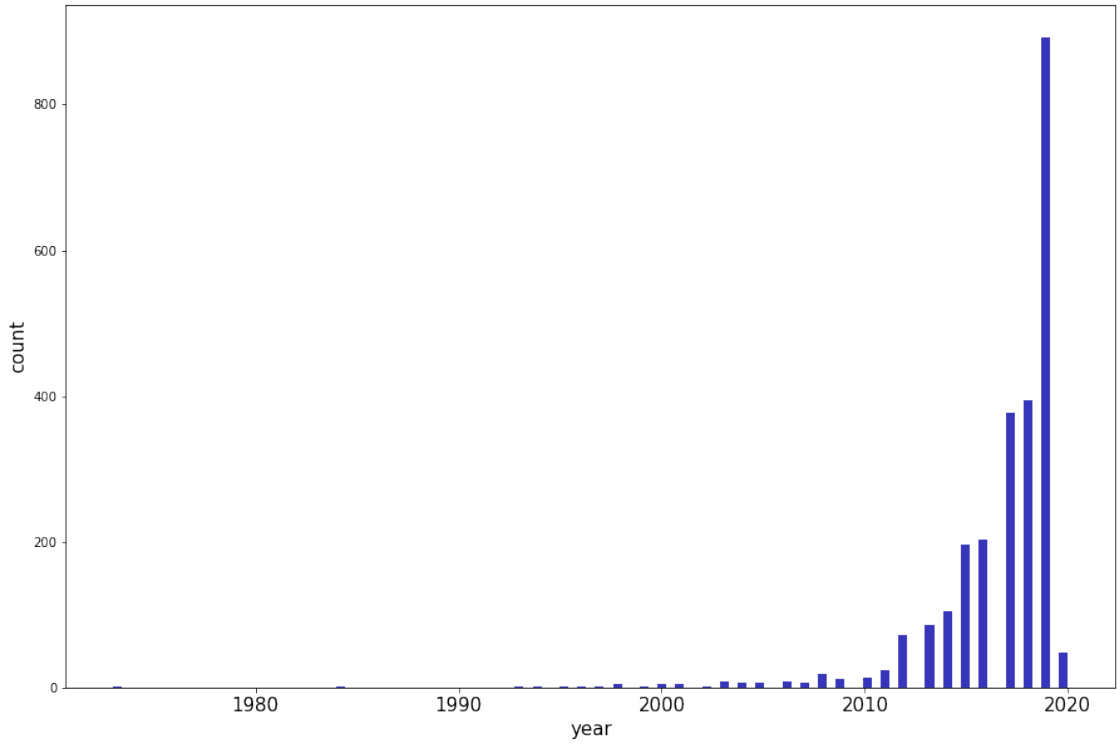
[46]: 2.3414478553472264

[47]:
```
# histogram of 'year'
```

[48]:
```
f = plt.figure()
f.set_figwidth(15)
f.set_figheight(10)

n, bins, patches = plt.hist(x=df['year'],bins='auto', color='#0504aa',
                            alpha=0.8, rwidth=10)
plt.rc('xtick', labelsize=15)
plt.rc('ytick', labelsize=10)
plt.xlabel('year', fontsize=15)
plt.ylabel('count', fontsize=15)
```

[48]: Text(0, 0.5, 'count')



[49]:
```
df.year.mean()
```

[49]: 2016.7142857142858

[50]:
```
df.year.median()
```

[50]: 2018.0

[51]:
```
stats.mode(df['year'])
```

[51]: ModeResult(mode=array([2019]), count=array([892]))

[52]:
```
df.year.var()
```

[52]: 11.85188150520371

[53]:
```
df.year.std()
```

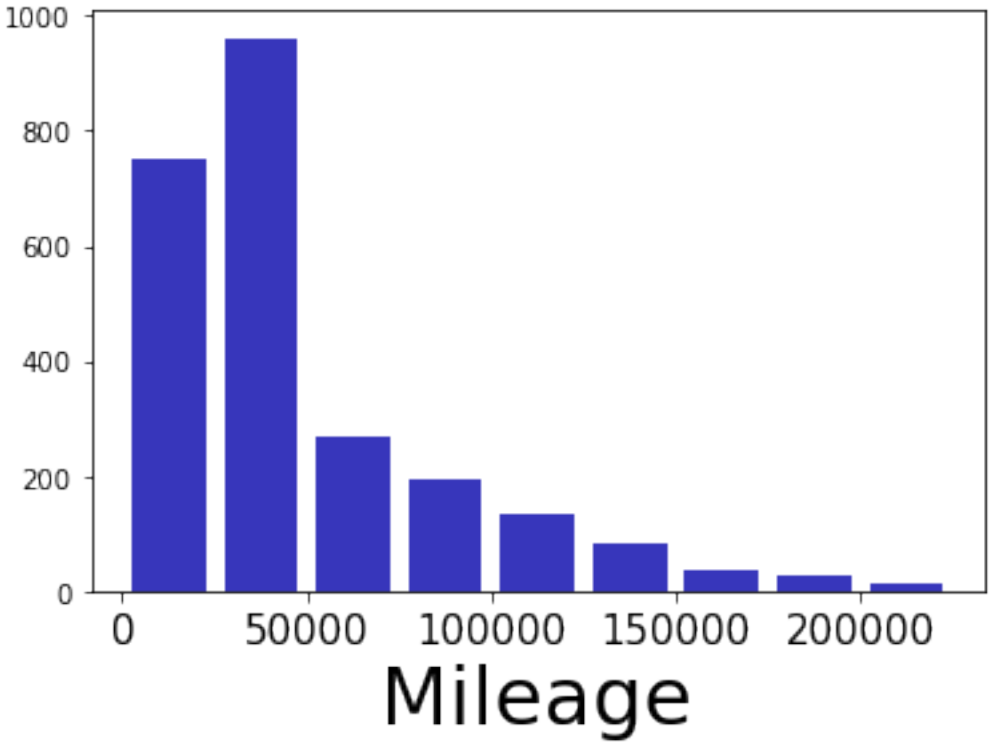
[53]: 3.442656170052959

[54]:
```
# histogram of 'mileage'
```

[55]:
```
f.set_figwidth(40)
f.set_figheight(10)
mileage_bins=np.arange(0, 250000, 25000)

n, bins, patches = plt.hist(x=df['mileage'],bins=mileage_bins, color='#0504aa',
                            alpha=0.8, rwidth=0.8)
plt.rc('xtick', labelsize=15)
plt.rc('ytick', labelsize=15)
plt.xlabel('Mileage', fontsize=30)
```

[55]:
```
Text(0.5, 0, 'Mileage')
```



[56]:
```
df.mileage.mean()
```

[56]:
```
52298.685474189675
```

[57]:
```
df.mileage.median()
```

[57]:
```
35365.0
```

[58]:
```
stats.mode(df['mileage'])
```

[58]:
```
ModeResult(mode=array([1.]), count=array([11]))
```

[59]:
```
df.mileage.var()
```

[59]:
```
3564748683.3886337
```

[60]:
```
df.mileage.std()
```

[60]:
```
59705.51635643589
```

From the above illustration of the data variables, the price ranges mostly between $1 to $40,000, while the most common value, which is the mode, is around $14,000. By looking at the distribution, the price is approximately right skewed with tails extends more at right than left. The moderately low amount of used car in luxury brand has caused the outliers, which we can't ignore, so I choose to keep the outliers. While when we look at the visualization of the brand,'ford', 'dodge', 'chevrolet', and 'nissan' has the most sells, among which 'ford' contributes almost half of the sells. Thers are some other car brand that has only a few amount of sell as outliers in the visualization. I'd like to keep them as well, since they have less sale, but we can't remove them. By looking at the histogram of year, variable 'year' is left skewed. Most used cars are registered after 2015, there are some from around 2000, and several from 1990s, which we need take them into consoderation. Mileage is right skewed, with most of the used car mileage are below 100000. While for variable color, they are randomly distributed, with white and black has the most in inventory.

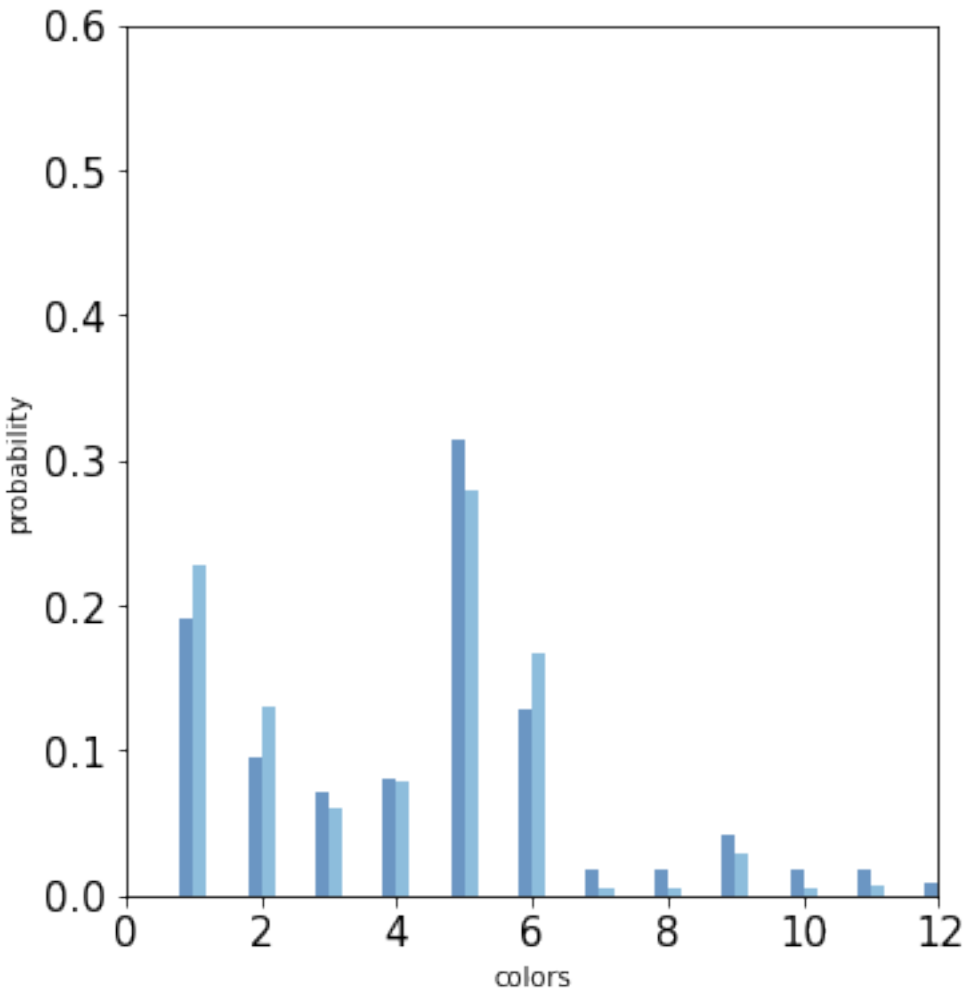
1.0.3 PMFs

[61]:
```
new = df[df.year > 2015]
old = df[df.year <= 2015]
```

[62]:
```
new_pmf_by_color = thinkstats2.Pmf(new['color_to_num'])
```

[63]:
```
old_pmf_by_color = thinkstats2.Pmf(old['color_to_num'])
```

[64]:
```
thinkplot.PrePlot(2, cols=2)
thinkplot.Hist(old_pmf_by_color, align='right', width=0.2)
thinkplot.Hist(new_pmf_by_color, align='left', width=0.2)
thinkplot.Config(xlabel='colors',ylabel='probability',axis=[0, 12, 0, 0.6])
```

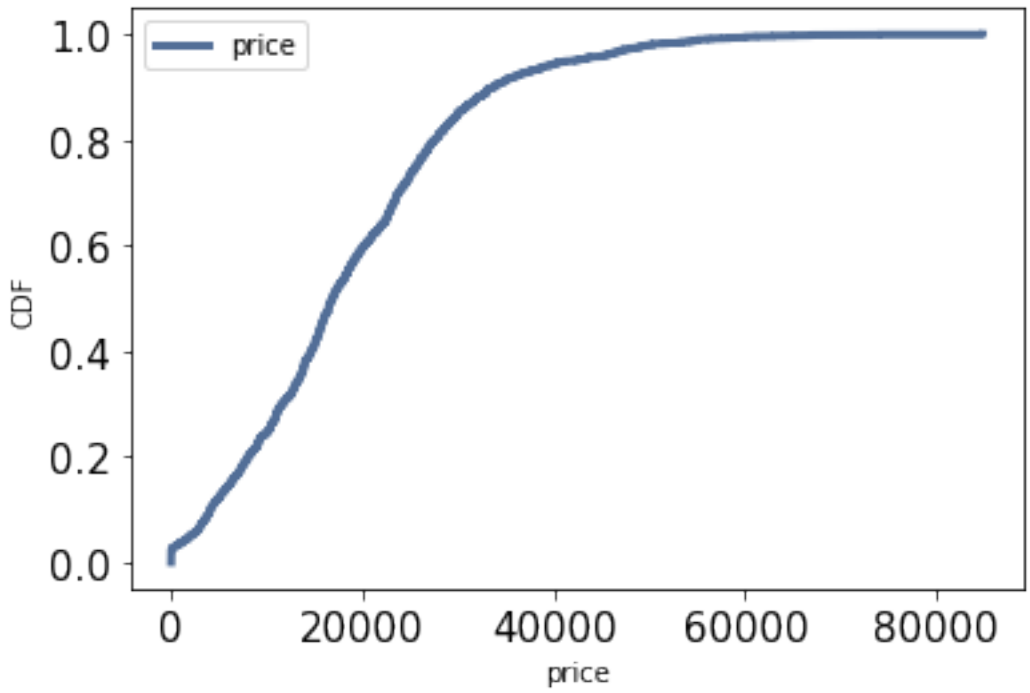


We could compare the distribution of colors in stock by the condition of the car to see which color has more inventory in used car market. I have set the registration year after 2015 new car, and in or before 2015 old car. By plotting the pmfs of old car and new car in different colors, we coul see that no matter old or new, the color "5" has the most inventory in used car market, which is "white" by referring the color number we had encoded.

1.0.4 CDFs

[65]:
```
price = df['price']
```

[66]:
```
cdf = thinkstats2.Cdf(price, label = "price")
thinkplot.Cdf(cdf)
thinkplot. Show(xlabel = 'price', ylabel = 'CDF')
```




```
<Figure size 576x432 with 0 Axes>
```

Above is the CDF of the price for used car sale. It looks like nearly 50% of useed car price are under $20000. The CDF also provides a visual presentation of the shape of the distribution. In this dataset, common values are below $3000, as from $30000 - $40000, there haven't been more values, while there are fewer values over the price of $40000, so the CDF above this price is flat.

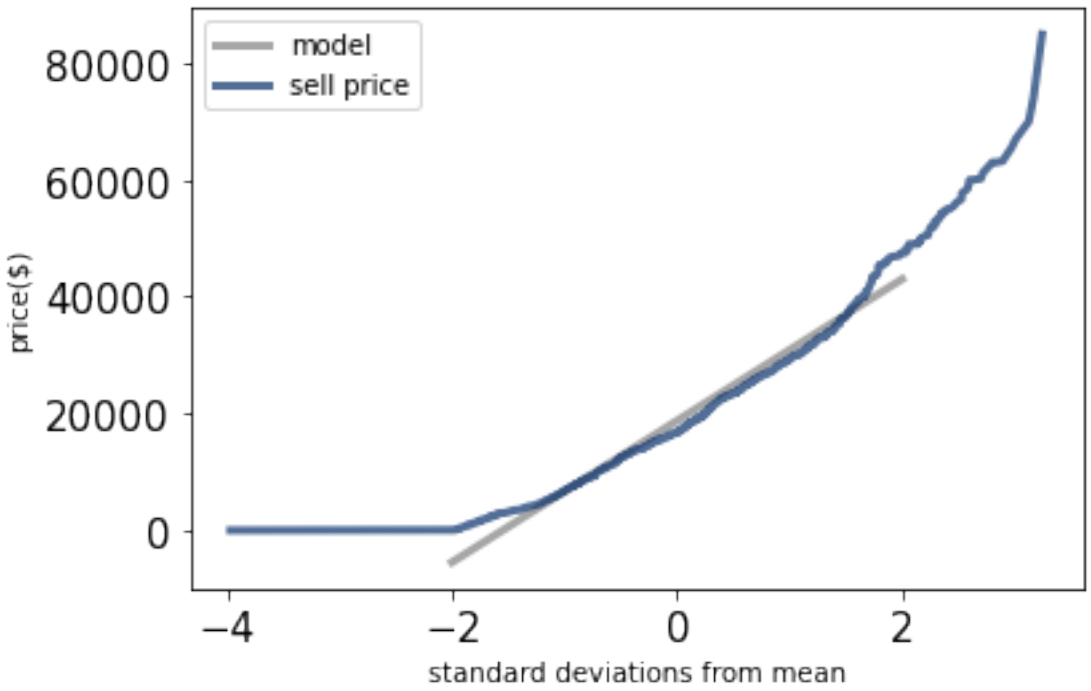
1.0.5 Plot 1 analytical distribution

[67]:
```
def EvalNormalCdf(x, mu=0, sigma=1):
    return scipy.stats.norm.cdf(x, loc=mu, scale=sigma)
```

[68]:
```
def MakeNormalPlot(price):
    mean = price.mean()
    std = price.std()

    xs = [-2, 2]
    fxs, fys = thinkstats2.FitLine(xs, inter = mean, slope = std)
    thinkplot.Plot(fxs, fys, color = 'grey', label = 'model')

MakeNormalPlot(price)
xs, ys = thinkstats2.NormalProbability(price)
thinkplot.Plot(xs, ys, label = 'sell price')
#thinkplot.text(-3,70000,"xx")
thinkplot.Show(xlabel='standard deviations from mean',ylabel='price($)')
```




```
<Figure size 576x432 with 0 Axes>
```

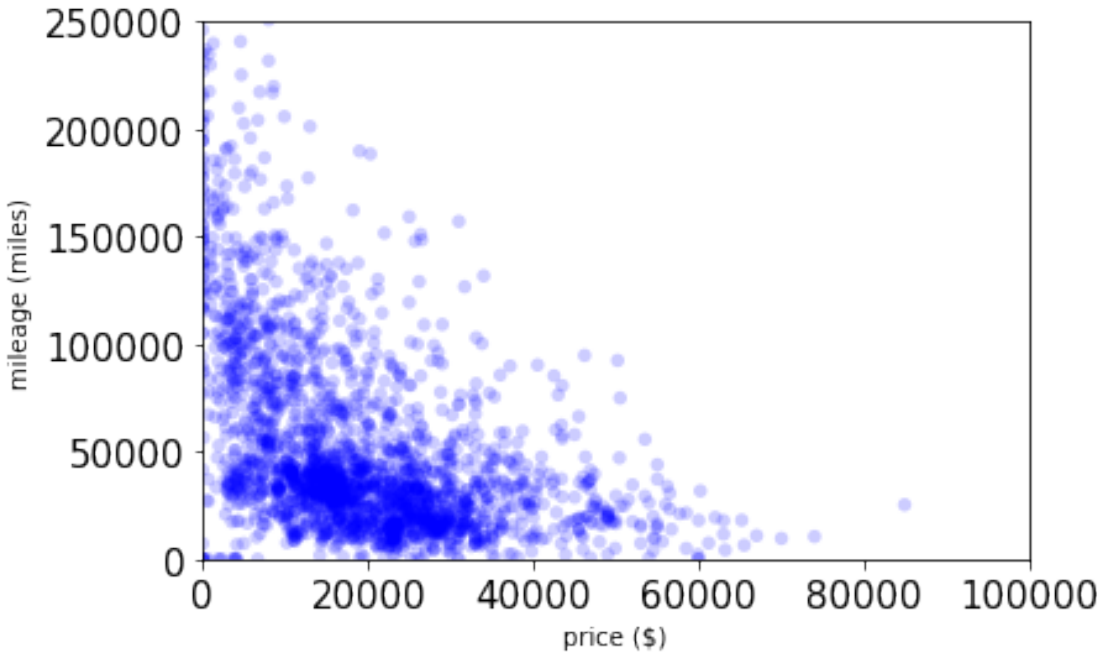
The inverted normal probability distribution curve indicates the distribution of price is right skewed, that's as what we have seen from the histogram of price. The mean, which the standard deviation of 0 is about $ 19000, meanwhile most of the sell price is under $40000.

1.0.6 Create two scatter plots comparing two variables

[69]:
```
# make scartter plots betweeen price and mileage.
```

[70]:
```
price, mileage = df.price, df.mileage
```

[71]:
```
thinkplot.Scatter(price, mileage, alpha = 0.2)
thinkplot.Show(xlabel = 'price ($)',
               ylabel = 'mileage (miles)',
               axis = [0, 100000, 0, 250000])
```




```
<Figure size 576x432 with 0 Axes>
```

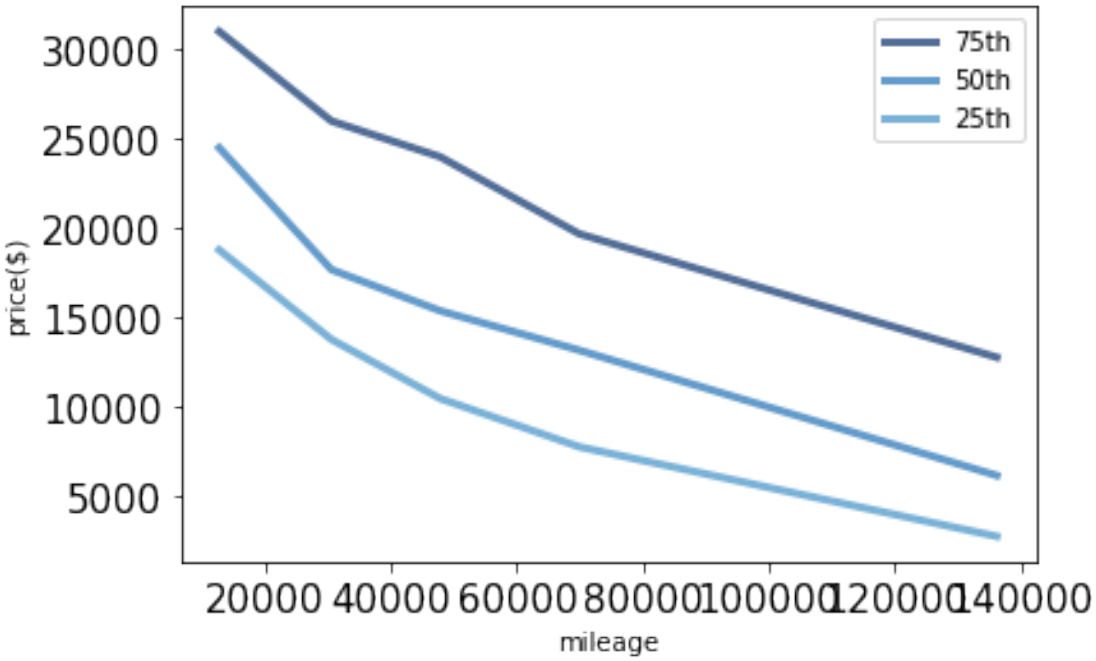
[72]:
```
# correlation and causation
```

[73]:
```
df = df.dropna(subset = ['mileage', 'price'])
bins = np.arange(0, 100000, 20000)
indices = np.digitize(df.mileage, bins)
groups = df.groupby(indices)
```

[74]:
```
mileage = [group.mileage.mean() for i, group in groups]
cdfs = [thinkstats2.Cdf(group.price) for i, group in groups]
```

[75]:
```
for percent in [75, 50, 25]:
    price = [cdf.Percentile(percent) for cdf in cdfs]
    label = '%dth' % percent
    thinkplot.Plot(mileage, price, label = label)

thinkplot.Show(xlabel='mileage',ylabel='price($)')
```




```
<Figure size 576x432 with 0 Axes>
```

By looking at the scatter plot, we could simply see the correlation is roughly linear, as mileage goes up, price drops down. Also, by binning mileage, I plotted the percentile of price. The results shows that the relationship between mileage and price is linear for more than 99% of the data.

[76]:
```
# covariance, pearson's correlation and spearman's correlation
```

[77]:
```
def Cov(xs, ys, meanx=None, meany=None):
    xs = np.asarray(xs)
    ys = np.asarray(ys)

    if meanx is None:
        meanx = np.mean(xs)
    if meany is None:
        meany = np.mean(ys)

    cov = np.dot(xs-meanx, ys-meany) / len(xs)
    return cov
```

[78]:
```
def Corr(xs, ys):
    xs = np.asarray(xs)
    ys = np.asarray(ys)

    meanx, varx = thinkstats2.MeanVar(xs)
    meany, vary = thinkstats2.MeanVar(ys)
```

```
    corr = Cov(xs, ys, meanx, meany) / np.sqrt(varx * vary)
    return corr
```

[79]:
```
def SpearmanCorr(xs, ys):
    xranks = pd.Series(xs).rank()
    yranks = pd.Series(ys).rank()
    return Corr(xranks, yranks)
```

[80]:
```
Cov(price, mileage)
```

[80]: -220254387.08729172

[81]:
```
Corr(price, mileage)
```

[81]: -0.9533324072984439

[82]:
```
SpearmanCorr(price, mileage)
```

[82]: -1.0

1.0.7 Hypothesis test

[83]:
```
# print(df['zone'].unique())
```

[84]:
```
east = df.loc[df['zone'] == 'US/Eastern']
```

[85]:
```
west = df.loc[df['zone'] == 'US/Pacific']
```

[86]:
```
mountain = df.loc[df['zone'] == 'US/Mountain']
```

[87]:
```
central = df.loc[df['zone'] == 'US/Central']
```

[88]:
```
class DiffMeansPermute(thinkstats2.HypothesisTest):

    def TestStatistic(self, data):
        group1, group2 = data
        test_stat = abs(group1.mean() - group2.mean())
        return test_stat
    def MakeModel(self):
        group1, group2 = self.data
        self.n, self.m = len(group1), len(group2)
        self.pool = np.hstack((group1, group2))

    def RunModel(self):
        np.random.shuffle(self.pool)
        data = self.pool[:self.n], self.pool[self.n:]
        return data
```

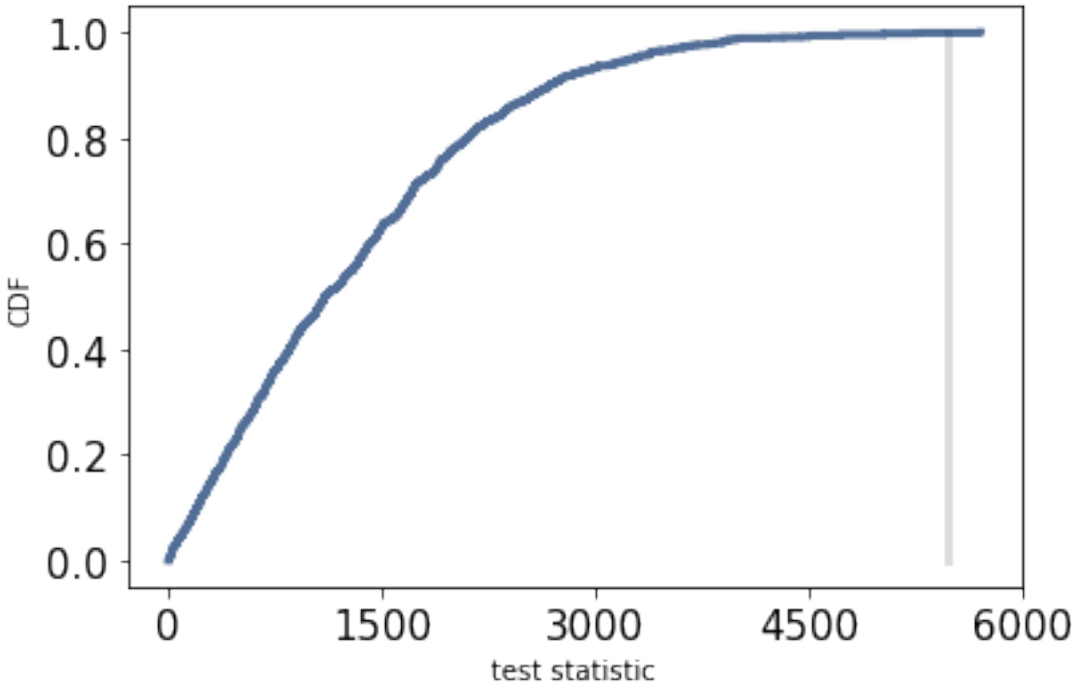
[89]:
```
data2 = mountain.price, central.price
```

[90]:
```
ht2 = DiffMeansPermute(data2)
```

[91]:
```
pvalue2 = ht2.PValue()
pvalue2
```

[91]:
```
0.001
```

[92]:
```
ht2.PlotCdf()
thinkplot.Show(xlabel='test statistic', ylabel='CDF',xticks =␣
↪[0,1500,3000,4500,6000])
```




```
<Figure size 576x432 with 0 Axes>
```

As we all know used car price might differ in different zones due to salary and cost of living. My hypothesis is the sell price in mountain zone and central zone has not much difference. The p value between price in moutain zone and central zone is 0.001. The CDF intersects the observed difference at 0.999, which is the complement of the p-value, 0.001. We expect to see an apparent differences as big as the observed effect of 5500 about 99.9% of the time.

1.0.8 Regression analysis

[93]:
```
formula = 'price ~ year + mileage + color_to_num'
results = smf.ols(formula, data=df).fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.211
Model:                            OLS   Adj. R-squared:                  0.210
Method:                 Least Squares   F-statistic:                     222.5
Date:                Sat, 18 Nov 2023   Prob (F-statistic):          6.54e-128
Time:                        17:20:37   Log-Likelihood:                -26746.
No. Observations:                2499   AIC:                         5.350e+04
Df Residuals:                    2495   BIC:                         5.352e+04
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept  -1.968e+06   1.58e+05    -12.479      0.000   -2.28e+06   -1.66e+06
year         986.0941     78.103     12.625      0.000     832.940    1139.248
mileage       -0.0479      0.004    -10.662      0.000      -0.057      -0.039
color_to_num  94.8817     92.721      1.023      0.306     -86.937     276.701
==============================================================================
Omnibus:                      526.342   Durbin-Watson:                   1.698
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1122.745
Skew:                           1.210   Prob(JB):                    1.58e-244
Kurtosis:                       5.220   Cond. No.                     5.81e+07
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 5.81e+07. This might indicate that there are
strong multicollinearity or other numerical problems.
```

For the regression analysis, I did the price with mileage, year, color and zone. The results above showed year and mileage are statistically significant of price, while color and zone are not as significant as year and mileage.

1.0.9 Conclusion

Here, 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. 12 features were assembled for each car in the dataset, several of them are very good candidate as explanatory variables to be used in the car price predictions. I recode some categorical attributes into numbers to faciliate a multi-variable regression analysis.Among them, the geographic variable

state were regrouped into 4 zones.

The overall average price is ~$18767 with a overall normal distribution (maybe a little bit right skewed), meanwhile, mileage, year of manufacturing, color as well as brand are the common factors to consider when car price need to evaluated.

In my explorator data analysis process, I did find out that used car price is very much related with the year and mileage of the car with negative correlation. Besides that, used car price has correlation with the state of the car's availability, which is car prices in east and pacific zone, the price is higher than that in central and mountain zone. I'm also interested in how car brand affected car price and how they're related.

The dataset is quite straightforward, the variables selected are all pretty relevant. I did the analysis using price, mileage, year, brand, color and state, which are considered to have some relations with used car prices. The assumptions I had made through my analysis seem reasonable.

On the other hand, although the data are all valid real market data entries, there is still opportunity for us to get a more comprehensive analysis. 1. the sample size is not very large, in this case, if there is a underline in group difference, it's may be difficult to detect with this sample size. 2. a few more explantory variables will be helpful for a more reliable regression, in current model, apparently year and milege will always get the highest weight which of course makes good practical sense, while in order to fundamentally reduce the MSE, more explantory variable will be the most helpful.

[ ]: