289Project - Used Car

December 19, 2017

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
import matplotlib
from sklearn import preprocessing
% matplotlib inline
```

1 Reading Data

In [2]:
```
data = pd.read_csv('./autos.csv', sep = ',', encoding = 'cp1252')
data.describe()
```

Out[2]:

	price	yearOfRegistration	powerPS	kilometer	\
count	3.715280e+05	371528.000000	371528.000000	371528.000000	
mean	1.729514e+04	2004.577997	115.549477	125618.688228	
std	3.587954e+06	92.866598	192.139578	40112.337051	
min	0.000000e+00	1000.000000	0.000000	5000.000000	
25%	1.150000e+03	1999.000000	70.000000	125000.000000	
50%	2.950000e+03	2003.000000	105.000000	150000.000000	
75%	7.200000e+03	2008.000000	150.000000	150000.000000	
max	2.147484e+09	9999.000000	20000.000000	150000.000000	

	monthOfRegistration	nrOfPictures	postalCode
count	371528.000000	371528.0	371528.00000
mean	5.734445	0.0	50820.66764
std	3.712412	0.0	25799.08247
min	0.000000	0.0	1067.00000
25%	3.000000	0.0	30459.00000
50%	6.000000	0.0	49610.00000
75%	9.000000	0.0	71546.00000
max	12.000000	0.0	99998.00000

1.0.1 Drop meaningless information and outliers

In [3]:
```
data.drop(['seller','offerType','dateCrawled','abtest','dateCreated','nrOfPictures','po
          ,axis = 'columns',inplace = True)
```

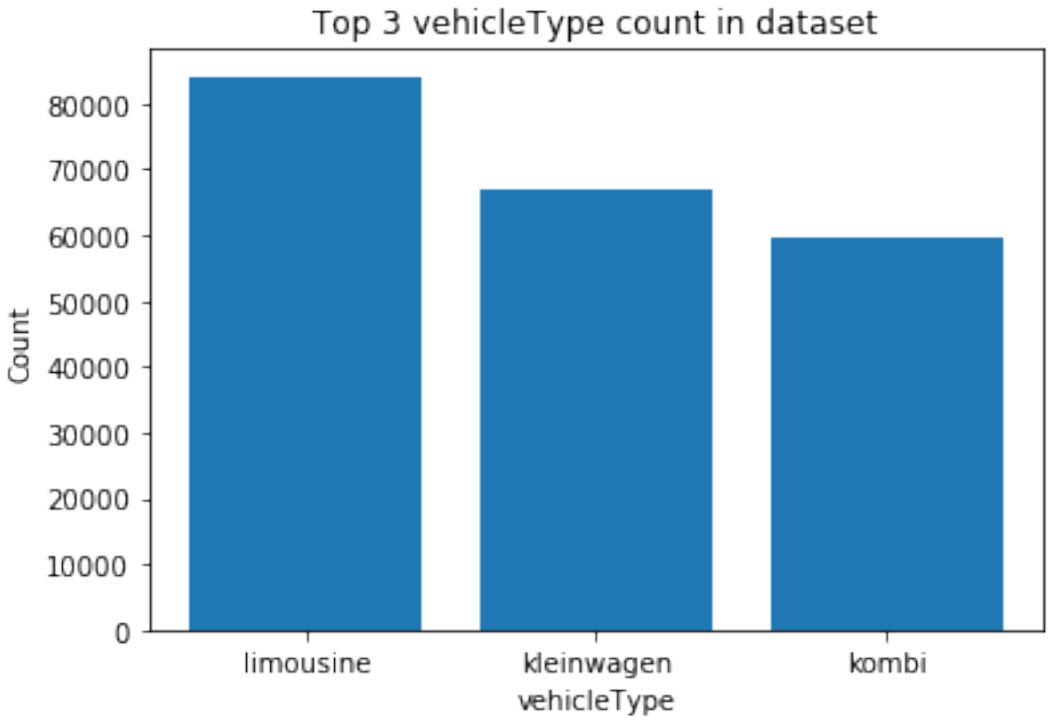
```
In [4]: # removing duplicate data
        column_name = data.columns.values.tolist()
        #print (column_name)
        dt = data.drop_duplicates(column_name)

In [5]: # removing outliers
        dt = dt[(dt.price>=200) & (dt.price <=200000) & (dt.yearOfRegistration>=1970) & (dt.yea
               & (dt.powerPS>=15) &(dt.powerPS <=500)]
```

1.0.2 Work on NAN value

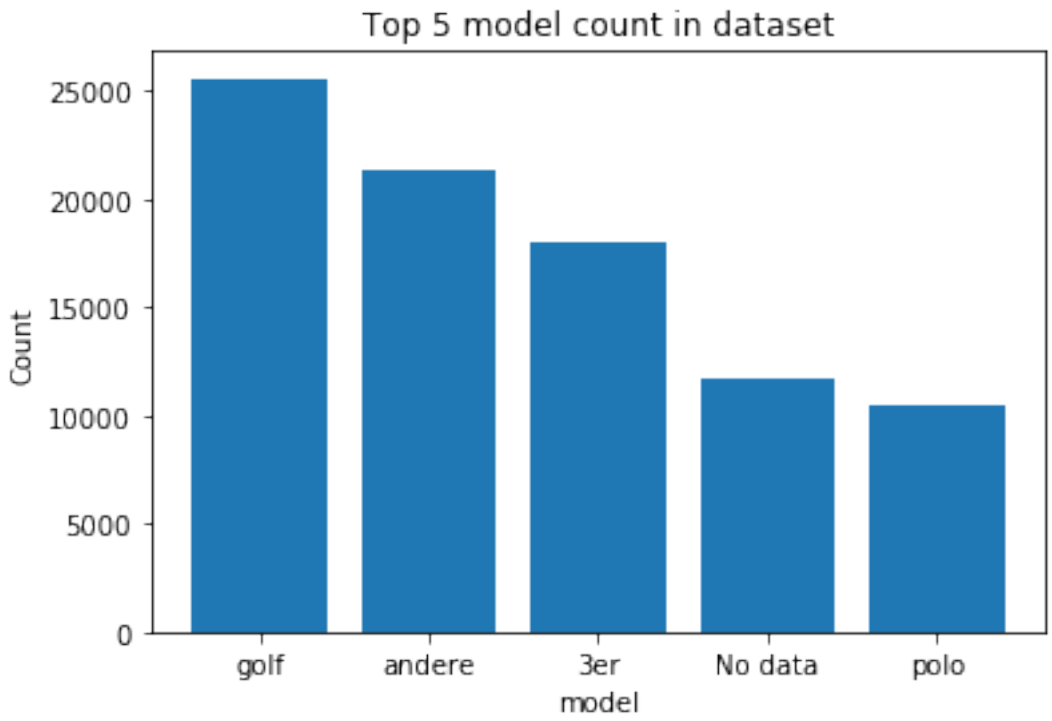
```
In [6]: dt.isnull().sum()

Out[6]: name                     0
        price                    0
        vehicleType          17613
        yearOfRegistration       0
        gearbox               5523
        powerPS                  0
        model                11701
        kilometer                0
        monthOfRegistration      0
        fuelType             17752
        brand                    0
        notRepairedDamage    44003
        dtype: int64

In [7]: dt['vehicleType'].fillna(value='No data', inplace=True)
        dt['gearbox'].fillna(value='No data', inplace=True)
        dt['model'].fillna(value='No data', inplace=True)
        dt['fuelType'].fillna(value='No data', inplace=True)
        dt['notRepairedDamage'].fillna(value='No data', inplace=True)
        dt.isnull().sum()

Out[7]: name                   0
        price                  0
        vehicleType            0
        yearOfRegistration     0
        gearbox                0
        powerPS                0
        model                  0
        kilometer              0
        monthOfRegistration    0
        fuelType               0
        brand                  0
        notRepairedDamage      0
        dtype: int64
```

2 Analyze Data

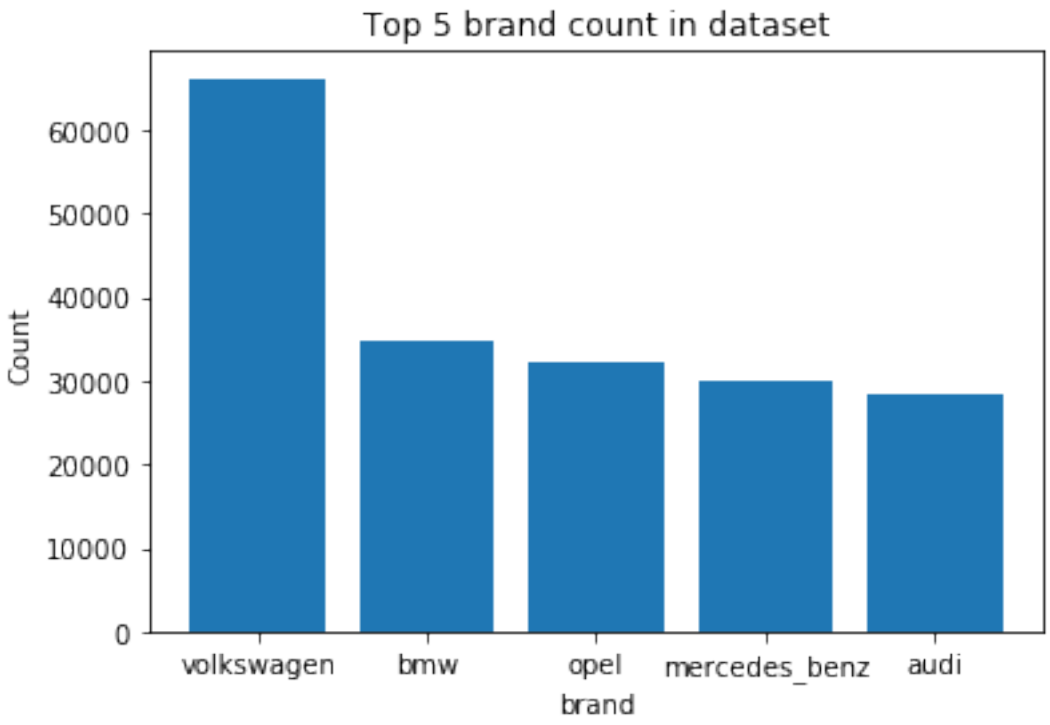
```
In [8]: vType = dt.groupby(by = 'vehicleType')['vehicleType'].count().sort_values(ascending = F
        show = 3
        x=range(show)
        plt.bar(x,vType.head(n=show))
        plt.xticks(x,vType.index)
        plt.xlabel('vehicleType')
        plt.ylabel('Count')
        plt.title ('Top 3 vehicleType count in dataset')
        plt.show()
```



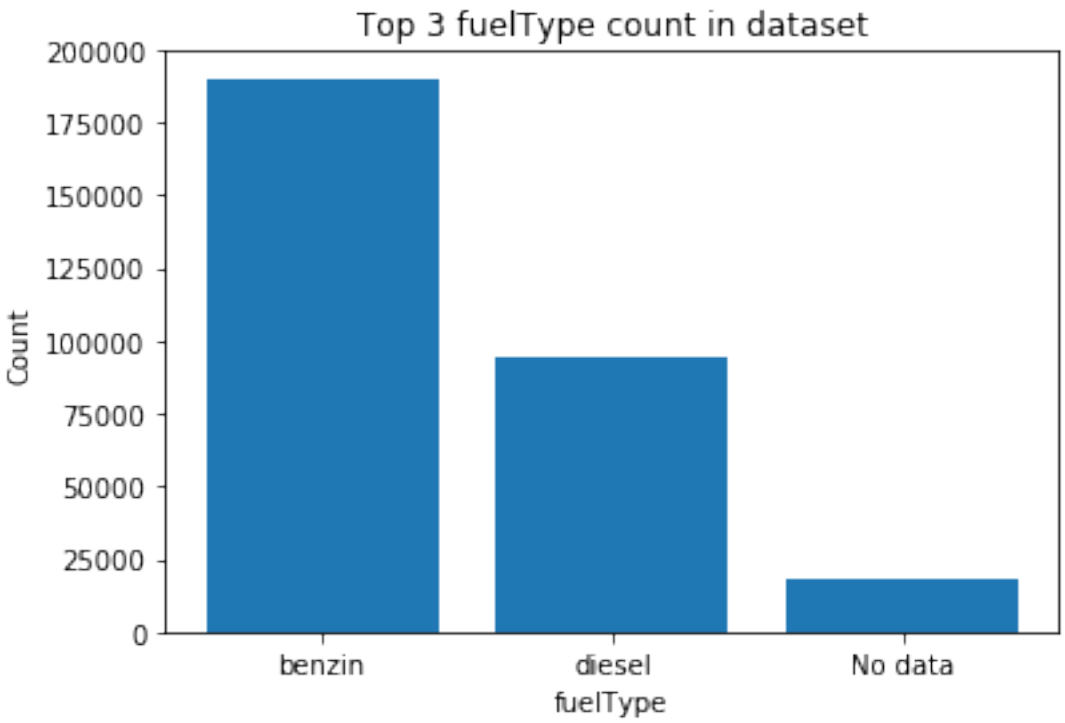

```
In [9]: info = 'model'
        vType = dt.groupby(by = info)[info].count().sort_values(ascending = False)
        show = 5
        x=range(show)
        plt.bar(x,vType.head(n=show))
        plt.xticks(x,vType.index)
        plt.xlabel(info)
        plt.ylabel('Count')
        plt.title ('Top 5 model count in dataset')
        plt.show()
```



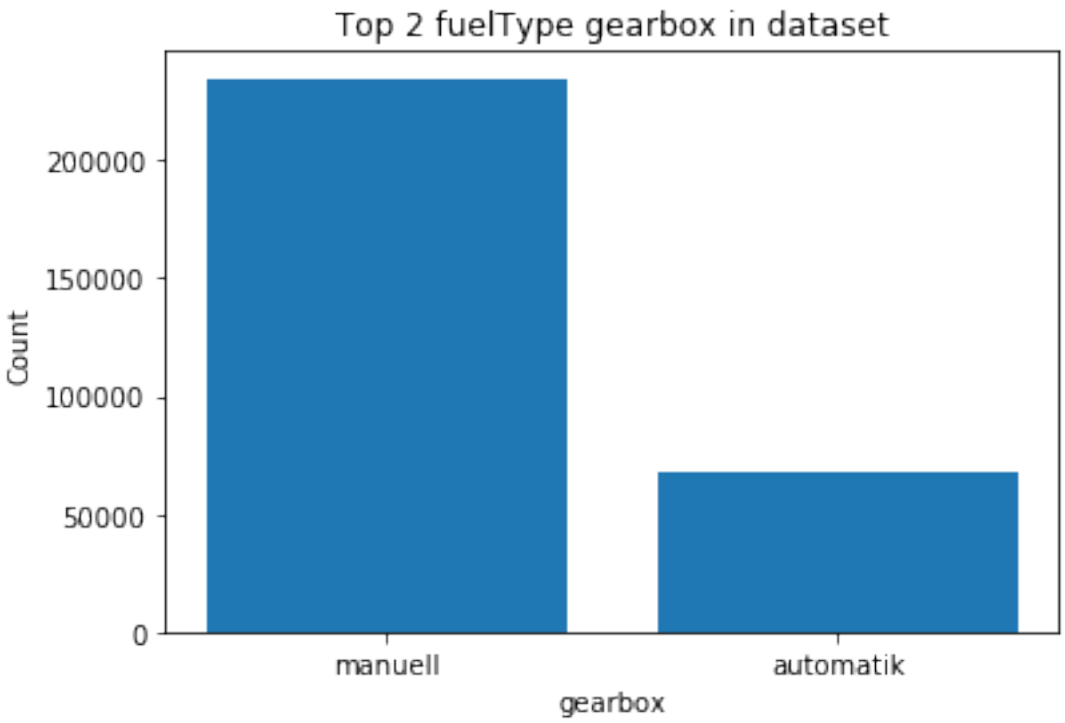

```
In [10]: info = 'brand'
         vType = dt.groupby(by = info)[info].count().sort_values(ascending = False)
         show = 5
         x=range(show)
         plt.bar(x,vType.head(n=show))
         plt.xticks(x,vType.index)
         plt.xlabel(info)
         plt.ylabel('Count')
         plt.title ('Top 5 brand count in dataset')
         plt.show()
```




```
In [11]: info = 'fuelType'
         vType = dt.groupby(by = info)[info].count().sort_values(ascending = False)
         show = 3
         x=range(show)
         plt.bar(x,vType.head(n=show))
         plt.xticks(x,vType.index)
         plt.xlabel(info)
         plt.ylabel('Count')
         plt.title ('Top 3 fuelType count in dataset')
         plt.show()
```




```
In [12]: info = 'gearbox'
         vType = dt.groupby(by = info)[info].count().sort_values(ascending = False)
         show = 2
         x=range(show)
         plt.bar(x,vType.head(n=show))
         plt.xticks(x,vType.index)
         plt.xlabel(info)
         plt.ylabel('Count')
         plt.title ('Top 2 fuelType gearbox in dataset')
         plt.show()
```



3 Organize and Prepare data

3.1 Change non-number features to class labels

```
In [13]: labels = ['gearbox', 'notRepairedDamage', 'model', 'brand', 'fuelType', 'vehicleType']
         les = {}
         dt['namelen'] = [min(70, len(n)) for n in dt['name']]   # change description to name
         for l in labels:
             les[l] = preprocessing.LabelEncoder()
             les[l].fit(dt[l])
             tr = les[l].transform(dt[l])
             dt.loc[:, l + '_feat'] = pd.Series(tr, index=dt.index)

         labeled = dt[ ['price','yearOfRegistration','powerPS','kilometer' ,'monthOfRegistrati
                        + [x+"_feat" for x in labels]]
         #labeled.head()
```

3.2 Prepare data for training

```
In [14]: Y = labeled['price']   #label
         X = labeled.drop(['price'], axis='columns', inplace=False)   # features

         Y = np.log1p(Y) # scaled price column
```

In [16]:
```
from sklearn.model_selection import cross_val_score, train_test_split

x_train, x_val, y_train, y_val = train_test_split(X, Y, test_size=0.25, random_state =
#print(x_train.shape, x_val.shape, y_train.shape, y_val.shape)
```

4 Train the model

4.0.1 Tune parameters by GridSearchCV

In [20]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
rf = RandomForestRegressor()

param = { "criterion" : ["mse"], "min_samples_leaf" : [4], "min_samples_split" : [6],

gs = GridSearchCV(estimator=rf, param_grid=param)
gs = gs.fit(x_train, y_train)
```

In [18]:
```
print(gs.best_score_)
print(gs.best_params_)
```

```
0.830467457395
{'criterion': 'mse', 'max_depth': 10, 'min_samples_leaf': 3, 'min_samples_split': 3, 'n_estimat
```

4.1 Train the model

In [26]:
```
bp = gs.best_params_
forest = RandomForestRegressor(criterion=bp['criterion'],
                               min_samples_leaf=bp['min_samples_leaf'],
                               min_samples_split=bp['min_samples_split'],
                               max_depth=bp['max_depth'],
                               n_estimators=bp['n_estimators'])
forest.fit(x_train, y_train)
print('Score: %f' % forest.score(x_val, y_val))
```

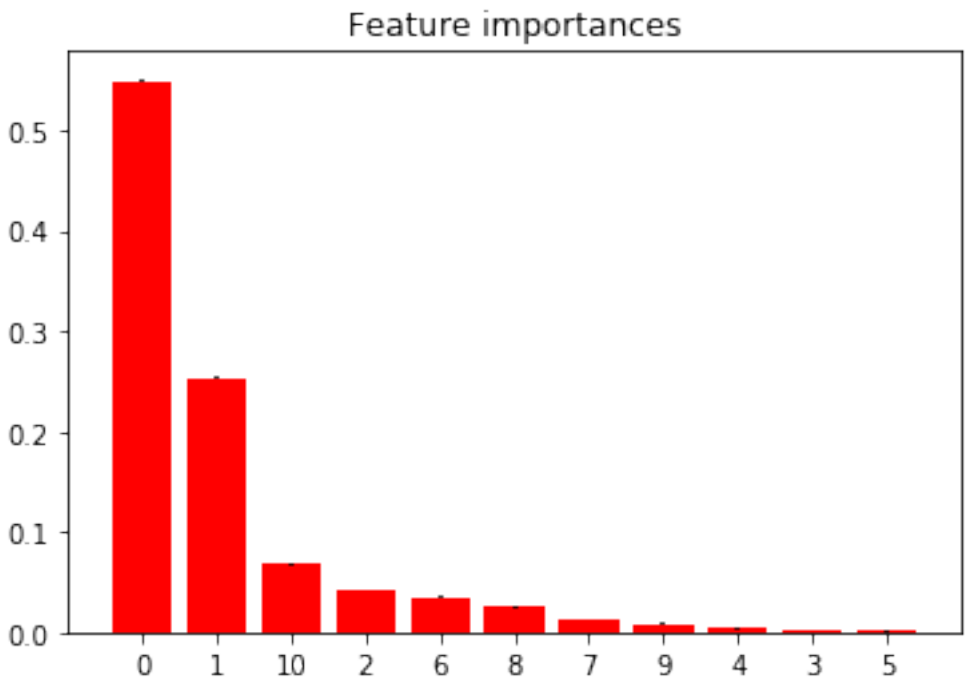
```
Score: 0.851633
```

4.2 Feature Importance

In [29]:
```
importances = forest.feature_importances_
std = np.std([tree.feature_importances_ for tree in forest.estimators_],
             axis=0)
indices = np.argsort(importances)[::-1]
# Print the feature ranking
print("Feature importances:")

for f in range(X.shape[1]):
```

```
            print("%d. feature %d (%f)" % (f + 1, indices[f], importances[indices[f]]))

        print(x_train.columns.values)

        # Plot the feature importances of the forest
        plt.figure()
        plt.title("Feature importances")
        plt.bar(range(X.shape[1]), importances[indices],
               color="r", yerr=std[indices], align="center")
        plt.xticks(range(X.shape[1]), indices)
        plt.xlim([-1, X.shape[1]])
        plt.show()
```

```
Feature importances:
1. feature 0 (0.548972)
2. feature 1 (0.252744)
3. feature 10 (0.067629)
4. feature 2 (0.042471)
5. feature 6 (0.034936)
6. feature 8 (0.025168)
7. feature 7 (0.012907)
8. feature 9 (0.007973)
9. feature 4 (0.003718)
10. feature 3 (0.002646)
11. feature 5 (0.000836)
['yearOfRegistration' 'powerPS' 'kilometer' 'monthOfRegistration' 'namelen'
 'gearbox_feat' 'notRepairedDamage_feat' 'model_feat' 'brand_feat'
 'fuelType_feat' 'vehicleType_feat']
```



In [ ]: