Predicting the Used Car Price

From Craigslist

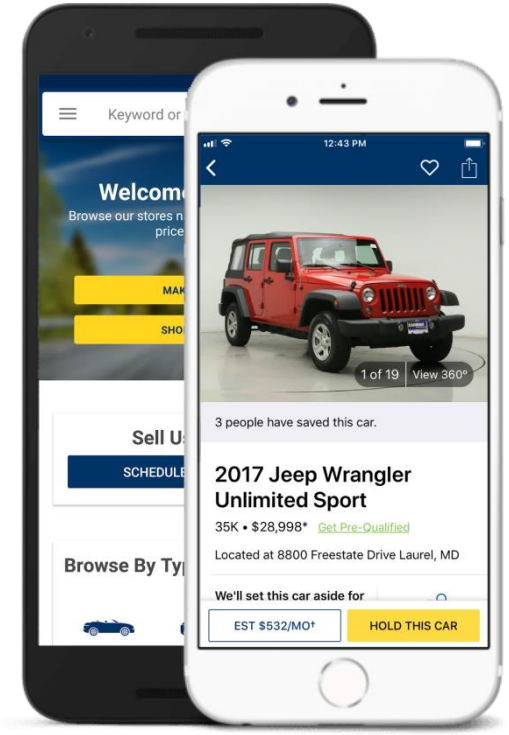
Yang Fei

Mentor: Kenneth Gil-Pasquel

Data Science Capstone Project 1, June 2020


What is the Target?
Physical Dealer
Price Prediction
predictions on the most popular Top 5 used car manufacturers on craigslist
Online Shopping

Who might cares?



Buyers

get a clear and concise price by observing an average standard.

Used Car Dealer

Make up their marketing plans by analyzing used car price trend and consumer psychology

Car Manufacturers

Used car price performance may affect their future development and sales strategies

Online APP Developer

Combine the prediction into their product to make it more competitive

Where is the data from?

Kaggle
https://www.kaggle.com/austinreese/craigslist-carstrucks-data

Craigslist
https://www.craigslist.org/
the world's largest used car platform.

Raw Data
509577 rows and 25 columns

Year
1990-2020

Format
csv

This data contains most all relevant information that Craigslist provides on car sales including columns like price, condition, manufacturer, latitude/longitude etc.

DATA CLEANING & WRANGLING

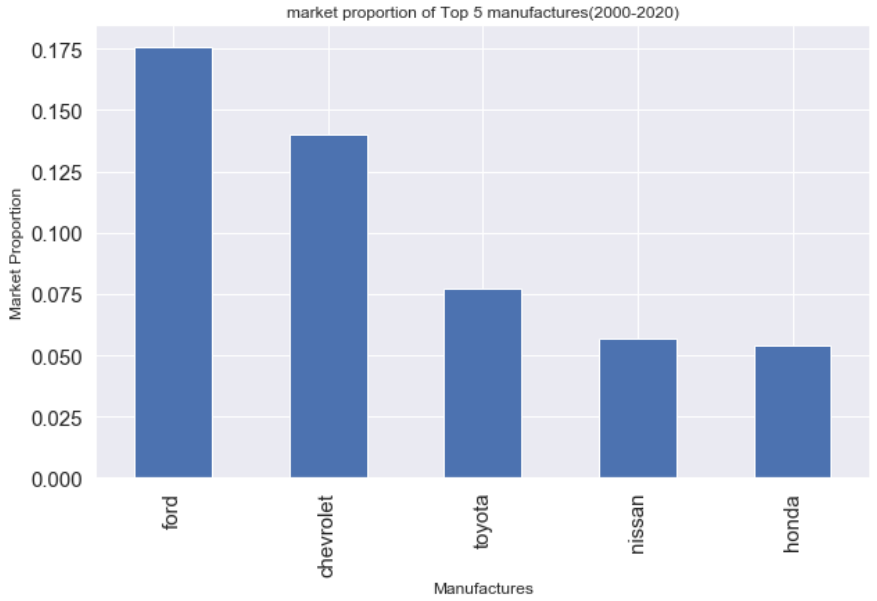
- Deal with missing and duplicated data
- Drop the fake data
- Detecting & Filtering Outliers
- Add more columns

Clean Data
485862 rows and 15 columns

Year
1990-2020

1. Add one additional column named 'age' which is calculated by 'year'
2. Add two columns name 'odometer_class' and 'price_class' for EDA Visualization.

What is the research object?



Research Period

Take the latest 20 years

Take the top 5 manufacturers which take the largest market proportion

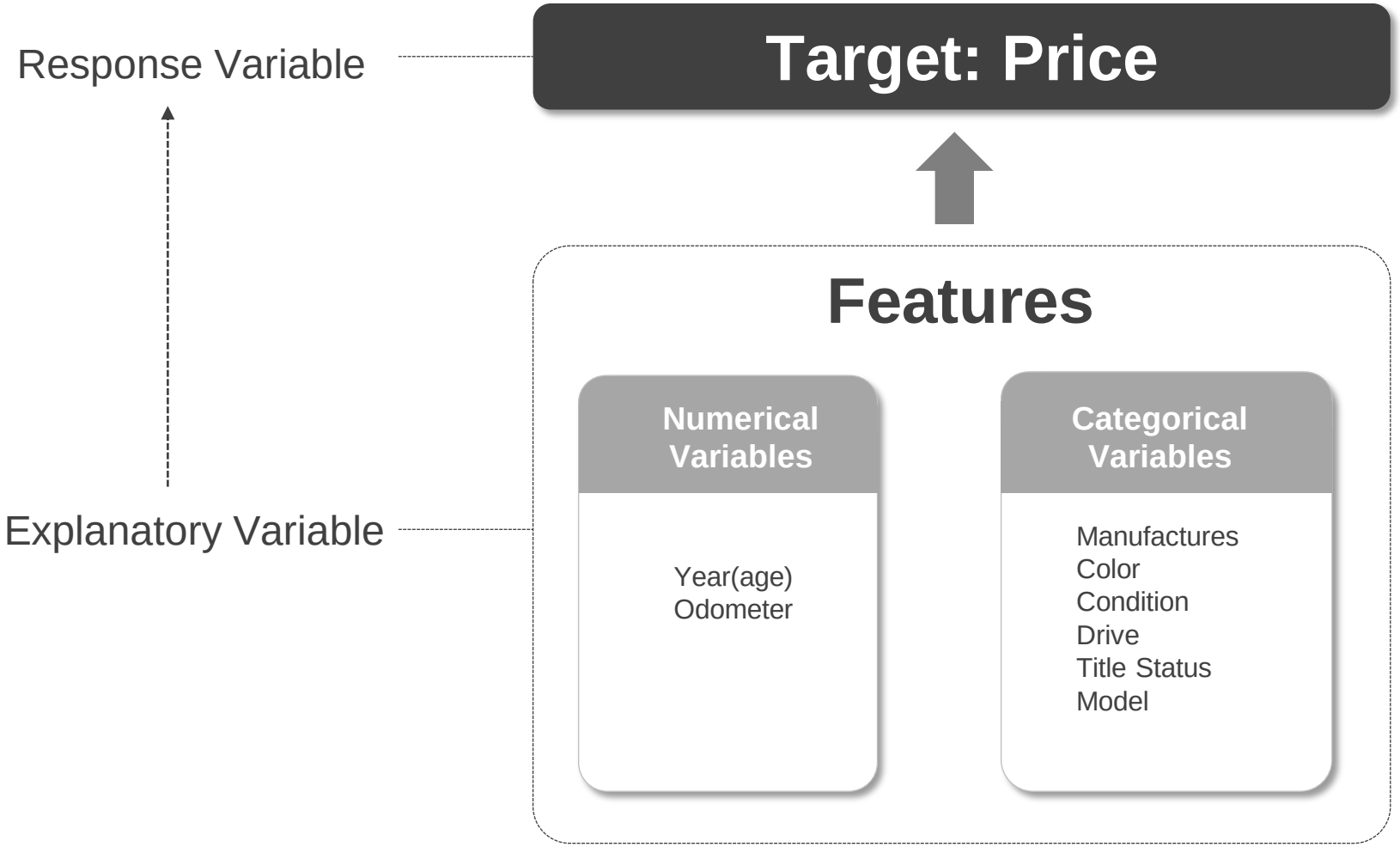
Manufacturers

Research Object

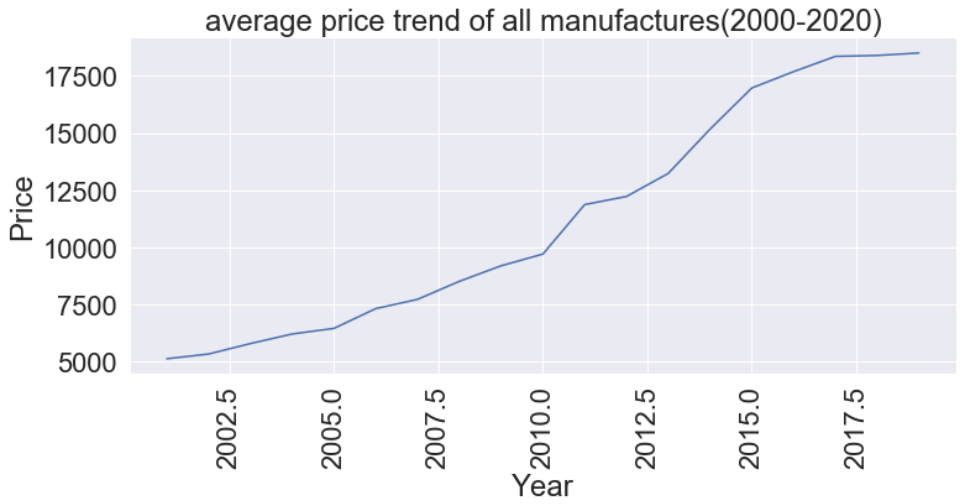
2000-2020

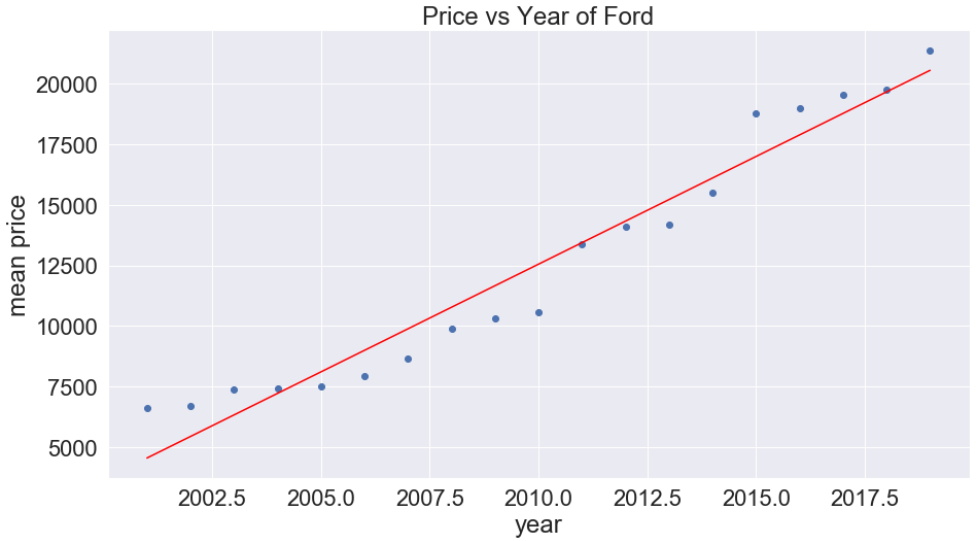
Ford
Chevrolet Toyota
Nissan
Honda

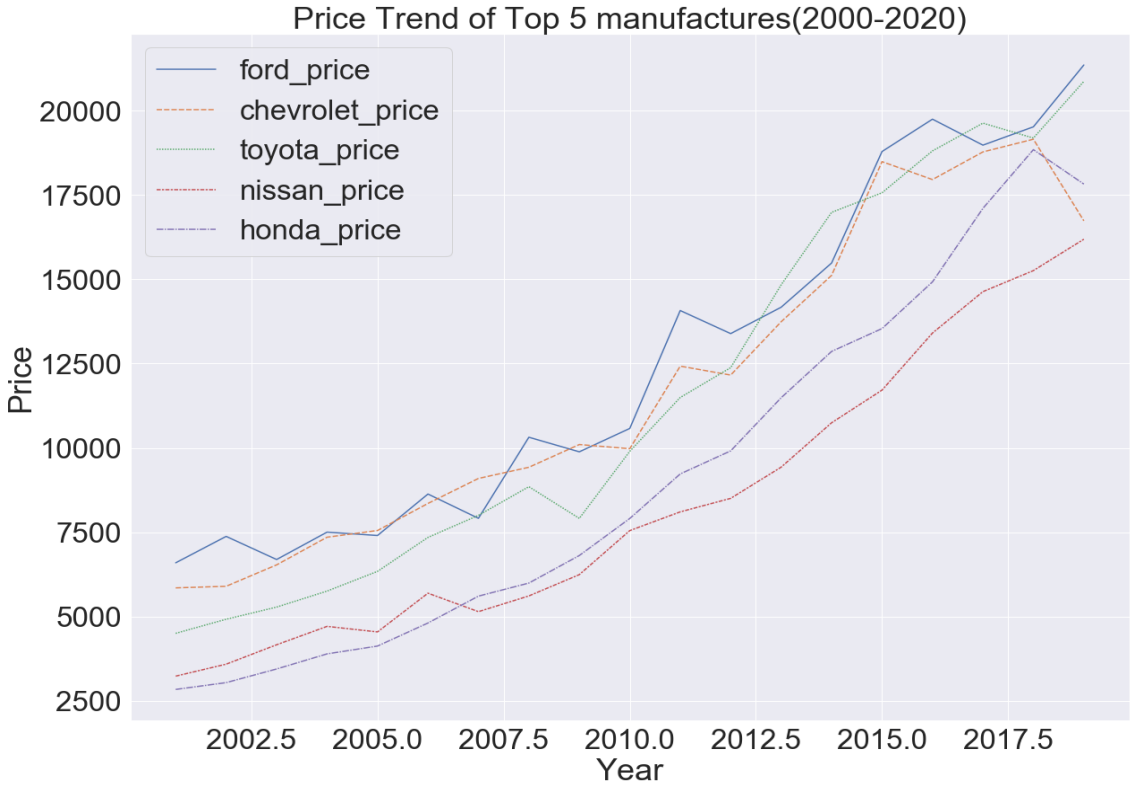
Data Exploration Analysis



Year and Price





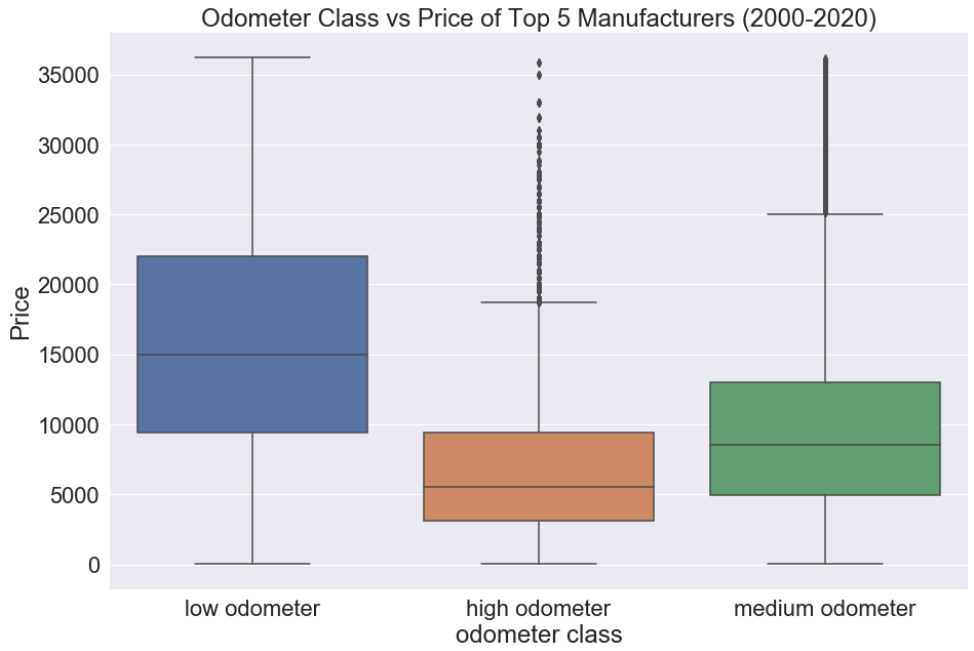


the average price of all used vehicles went up year by year. The increase speed is around 6% per year, which is a little higher than the inflation rate in America (3%). That's may be caused by online markets' developing. The used car price online is approaching to the physical dealer's gradually.

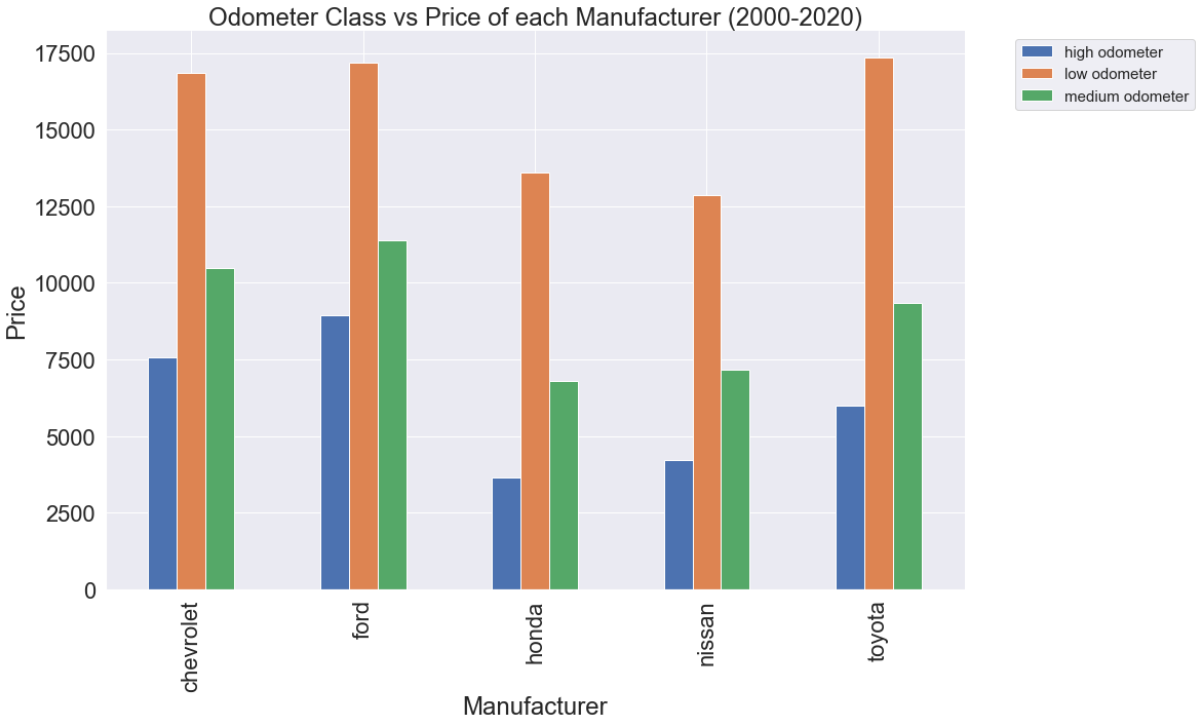
Pearson value is 0.5

There is a positive **linear** correlation between 'Year' and 'Price'.

Odometer and Price



Vehicles with low odometers usually have a higher mean price as well as a wider price range.

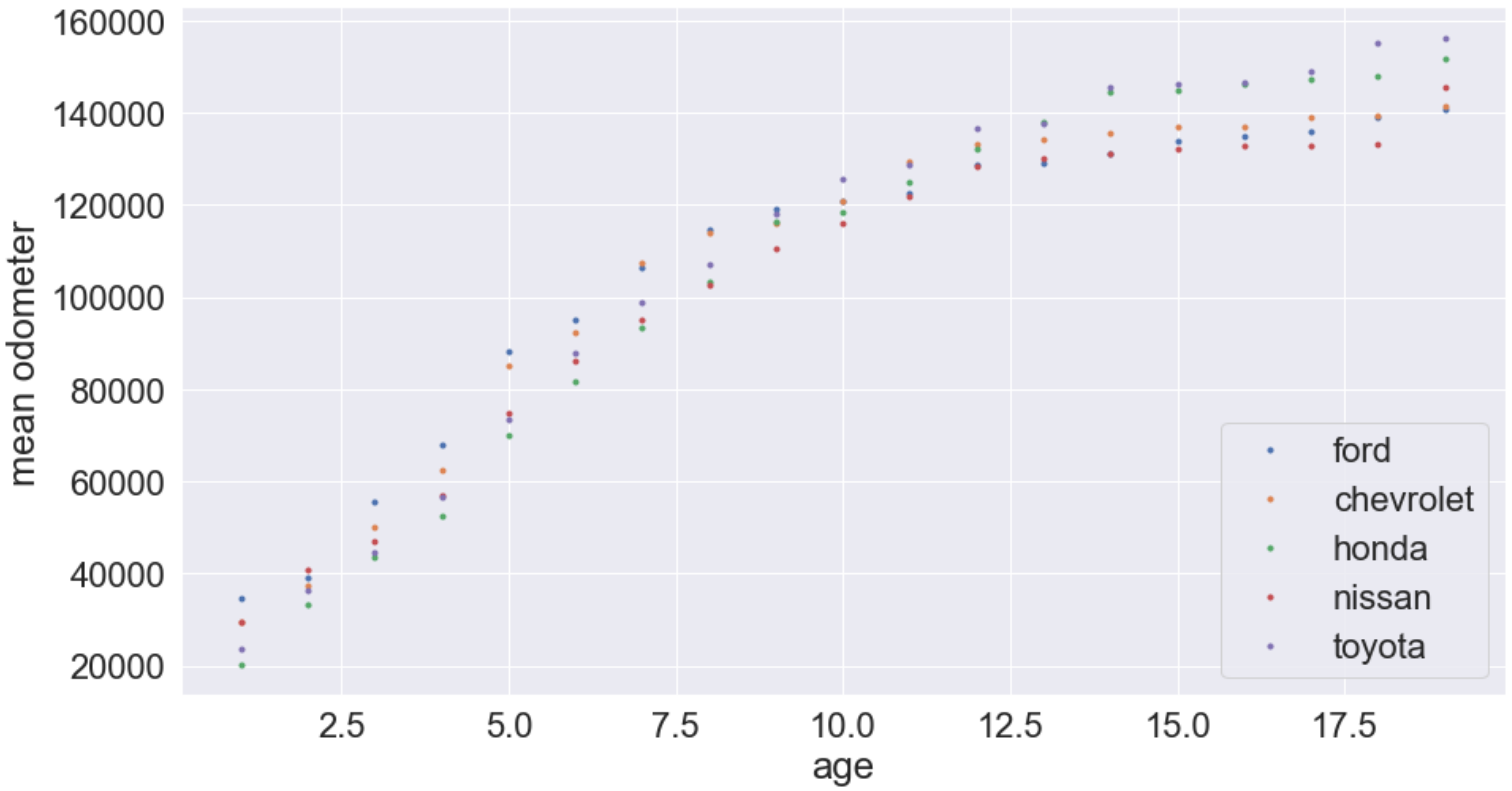


For different manufacturers, the relation between price and odometer is different. Toyota has the highest mean price of low kilometers(less than 10,000km). That's may be because of the statement that Japanese cars could held their value better than others in high odometer class.

Pearson value is 0.3

It's a positive correlation, but cannot be considered as a linear correlation.

Odometer and Year(Age)



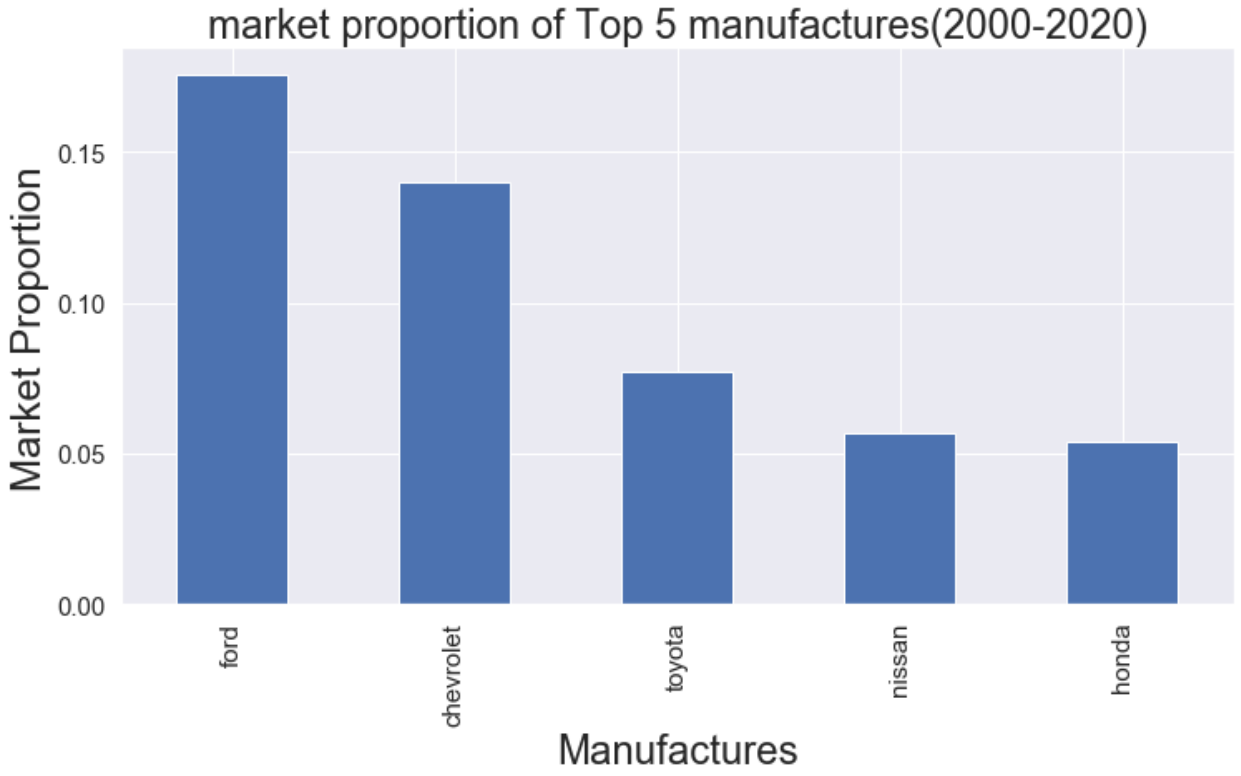
Pearson Value 0.5 indicate a positive linear relation between odometer and age.

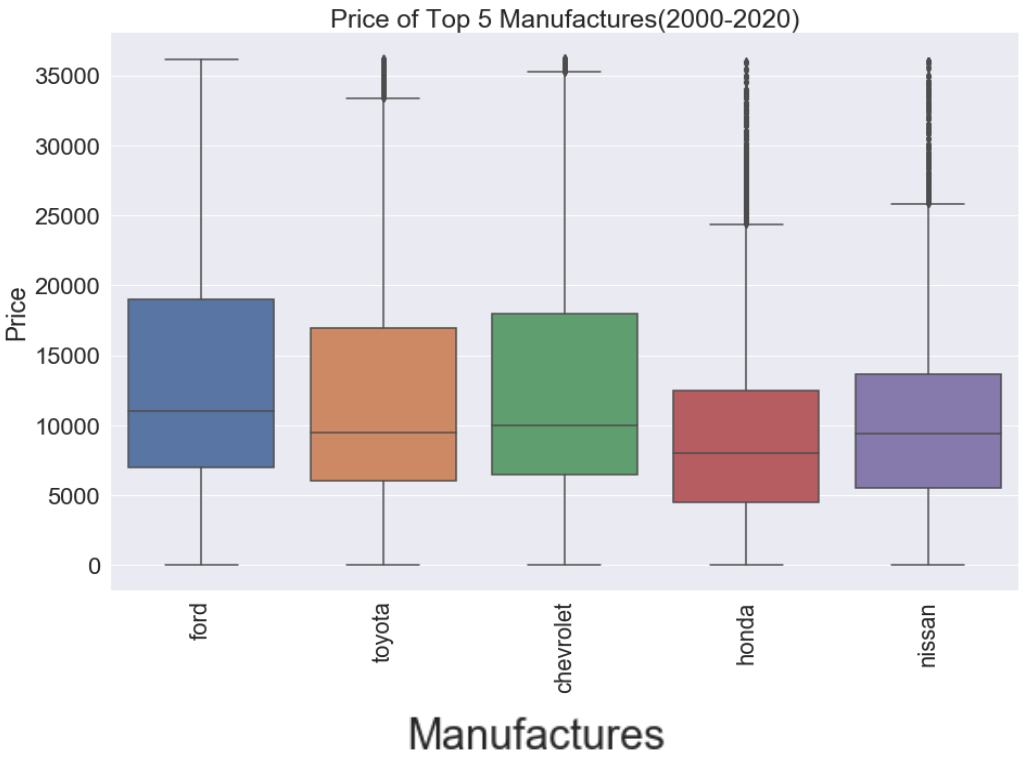
What is remarkable is that the growth rate of odometers slowed down significantly among the vehicles which are over 12.5 years old.

Manufacturers and Price

Combined with Market Proportion plot, it is obvious that their rankings are closely relevant.

Ford had the highest average price, followed by Chevrolet, Toyota, Nissan and Honda.



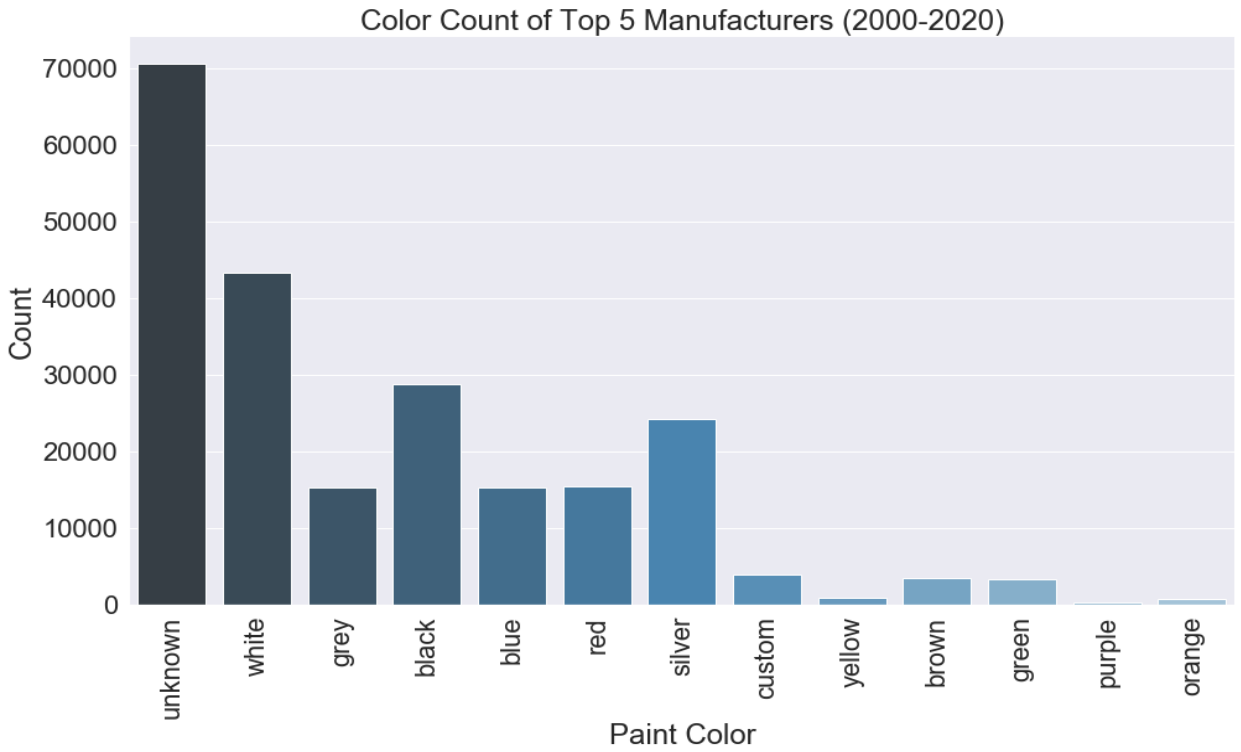


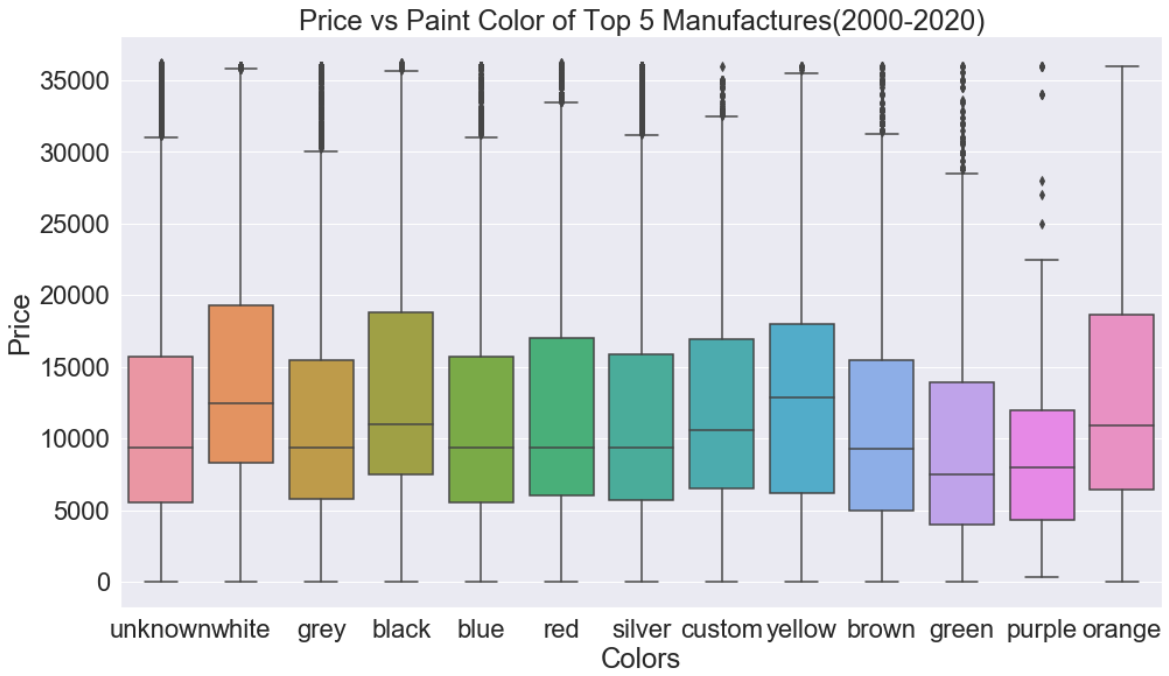
Colors and Price

There are more than 13 species of paint color cars on sale during the past 20 years.

Top 3 colors which have the largest market proportion are white, black and silver. Probably because orange is a popular color for customers, however without enough supply in market.

white, orange and black usually could be sold at a higher price than other colors.



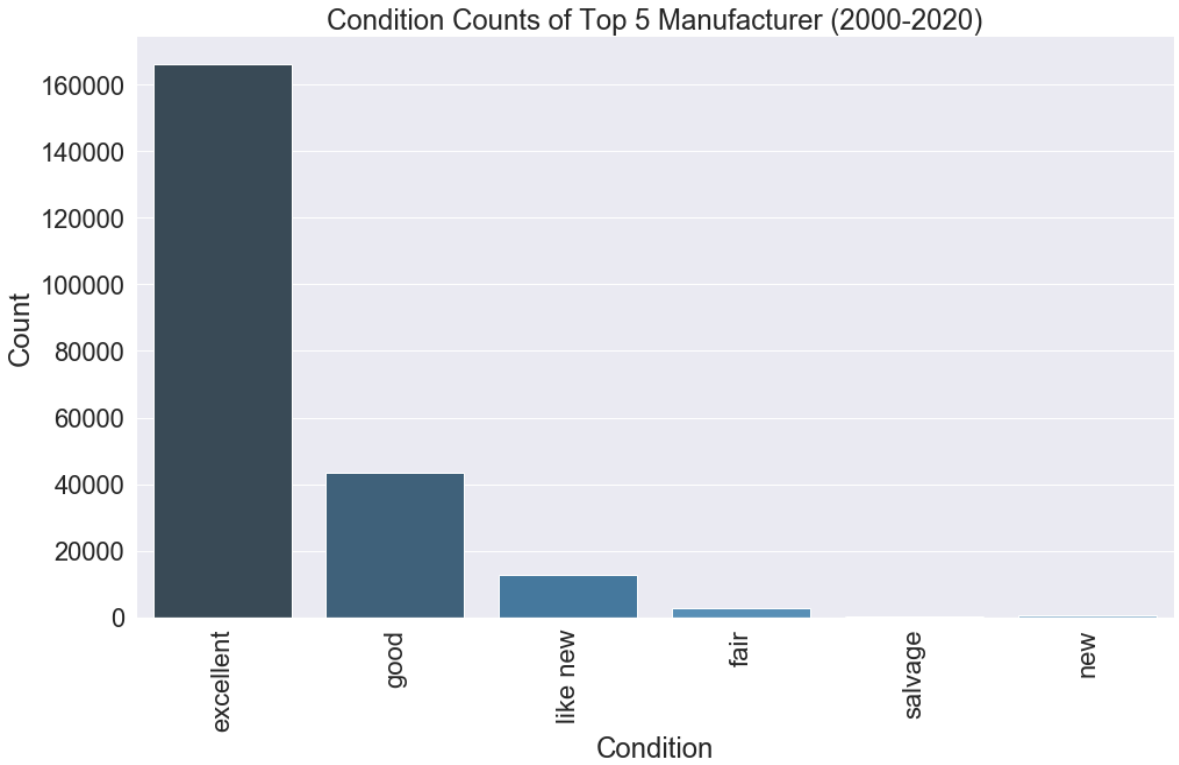


Condition and Price

The vehicles labeled as ‘Like new’ and ‘excellent’ condition have the highest median prices.

The ‘fair’ and ‘salvage’ ones have the smallest median prices as well as their price range.

Most vehicles on sale are described as **‘excellent condition’.**

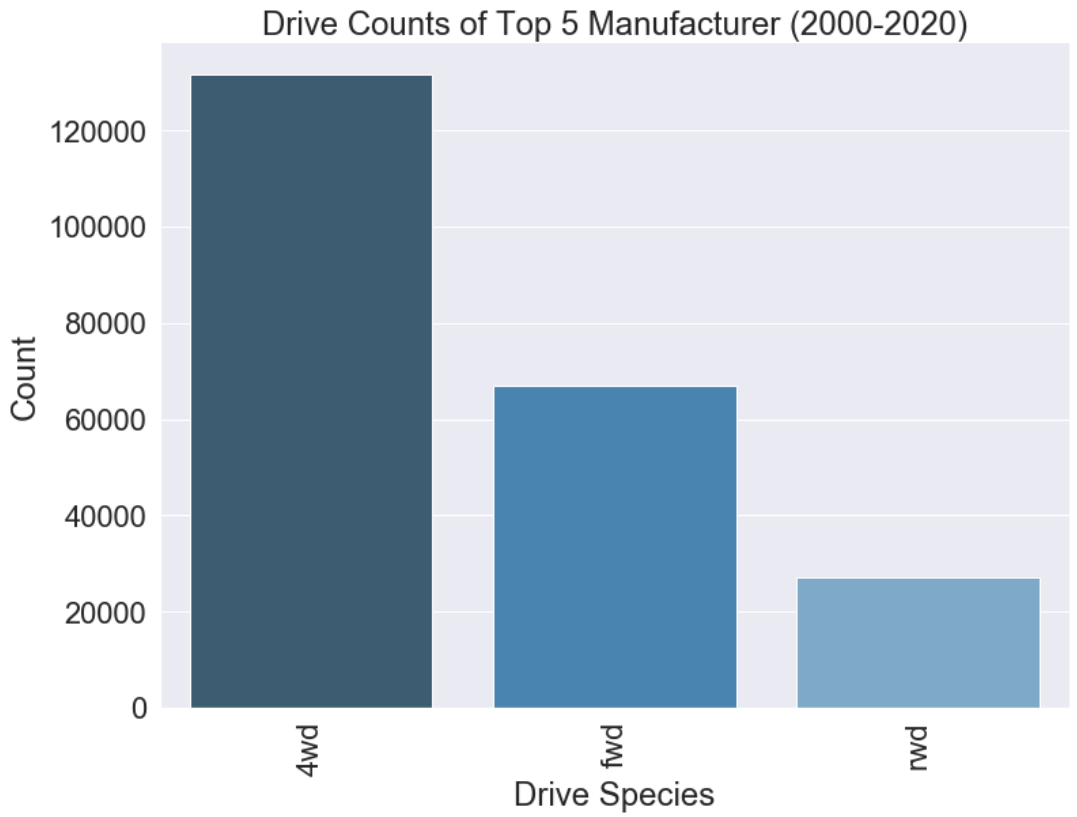


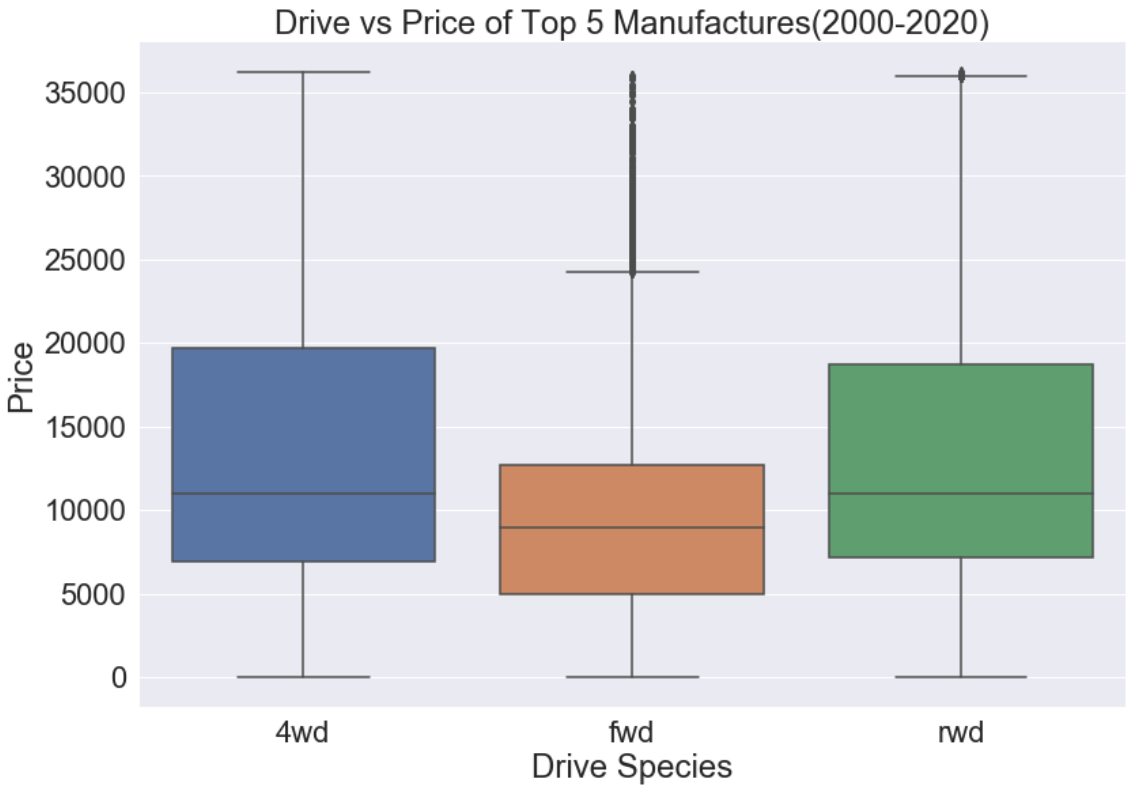
Drive and Price

4wd trucks seem to have a bigger share of the market, which could explain why Ford and Chevrolet are so popular.

4wd vehicles always have a higher median price than other two categories.

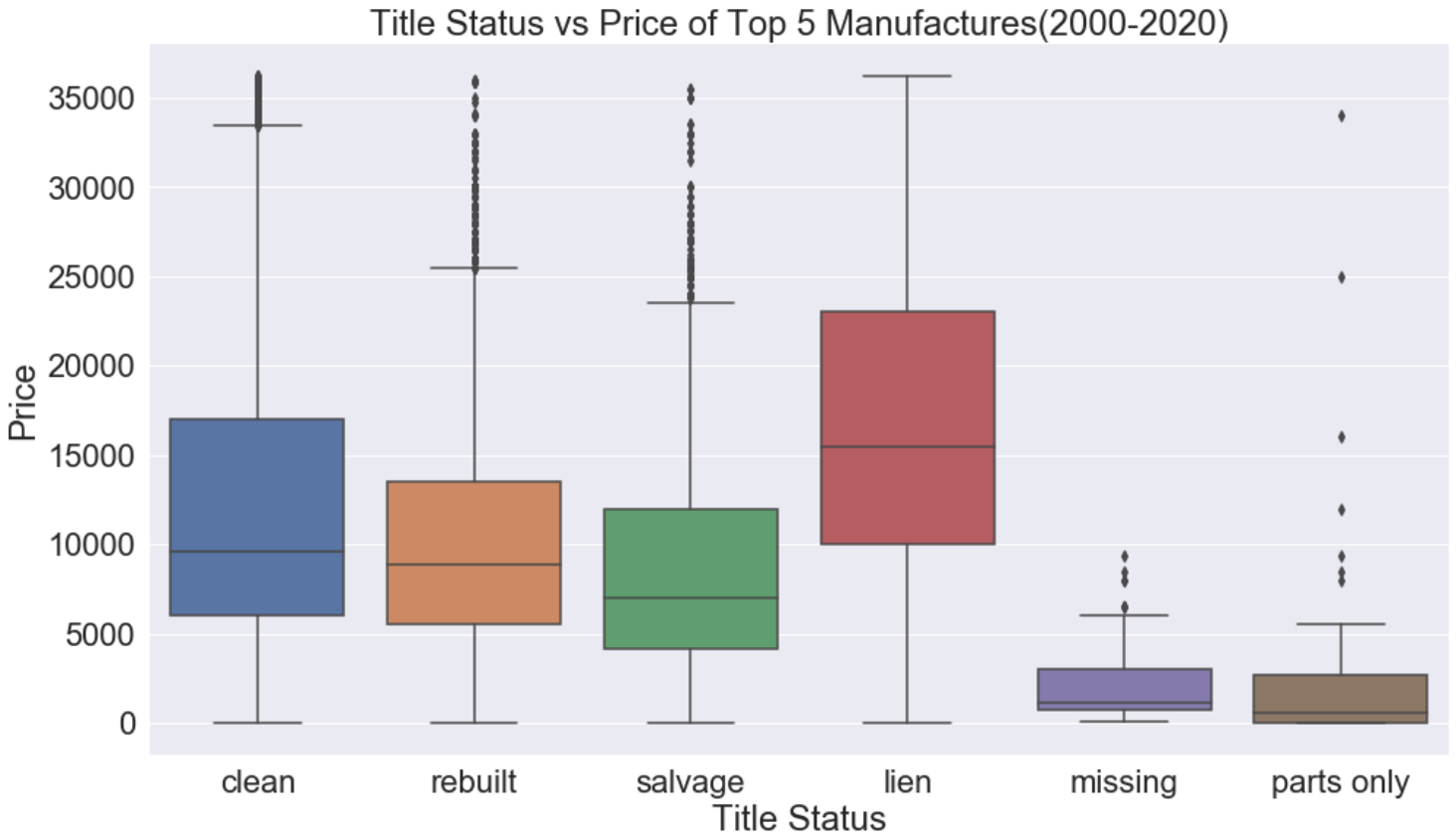
4wd shares the largest proportion of the market.





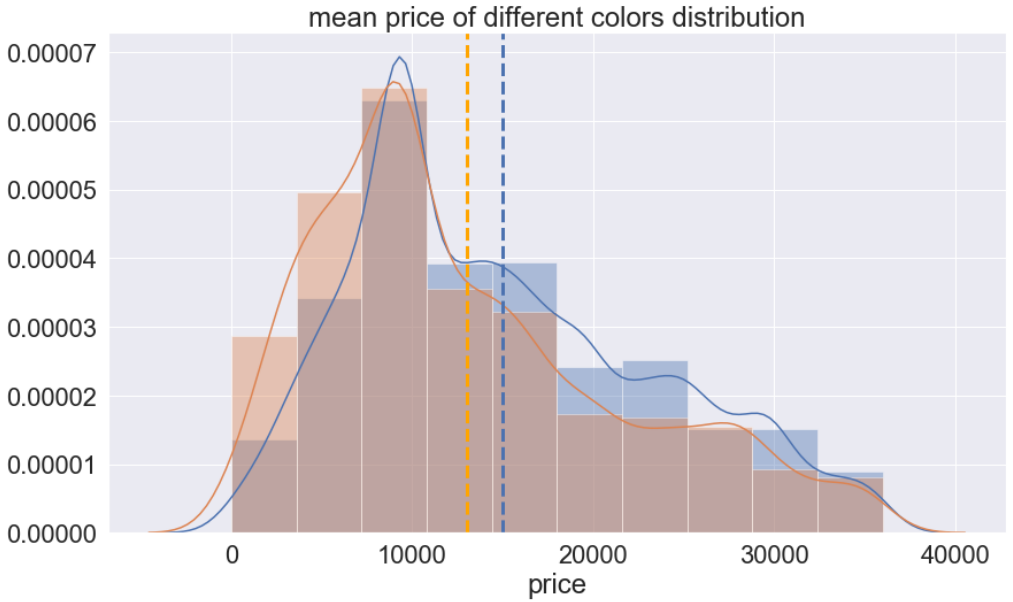
Title Status and Price

Lien vehicles have the highest median price of all, which also have the largest price range..



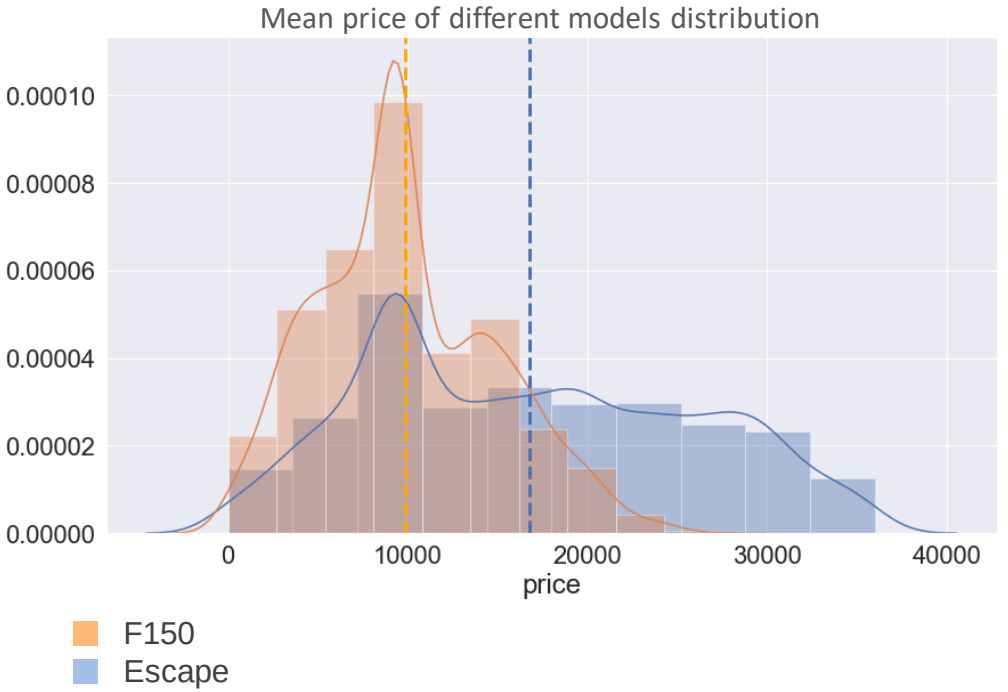
Statistical Data Analysis

Hypothesis Testing



H0: Samples from different colors share the same mean price
p-value is 0.039

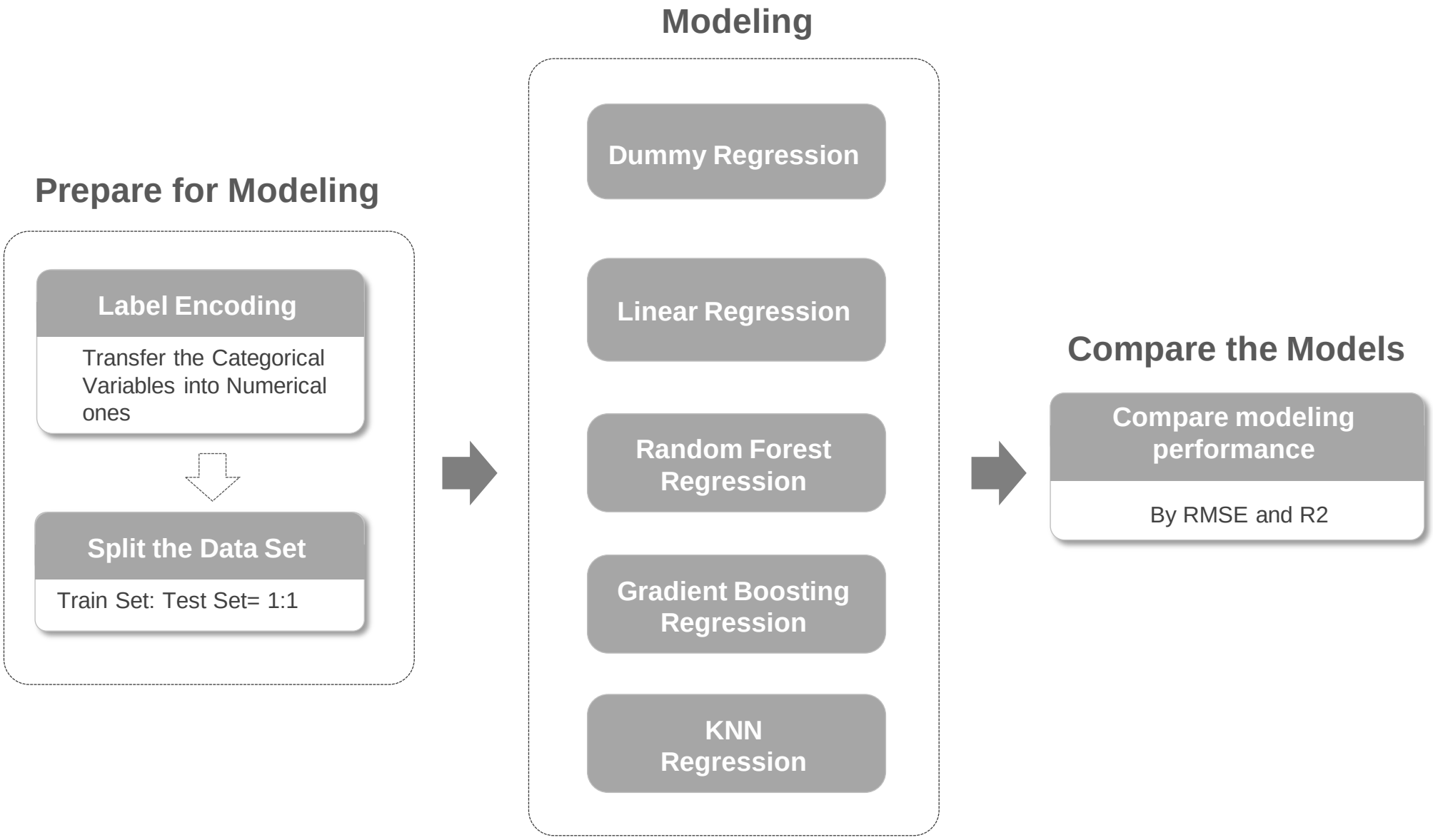
Reject H0: Samples from different colors share the different mean prices



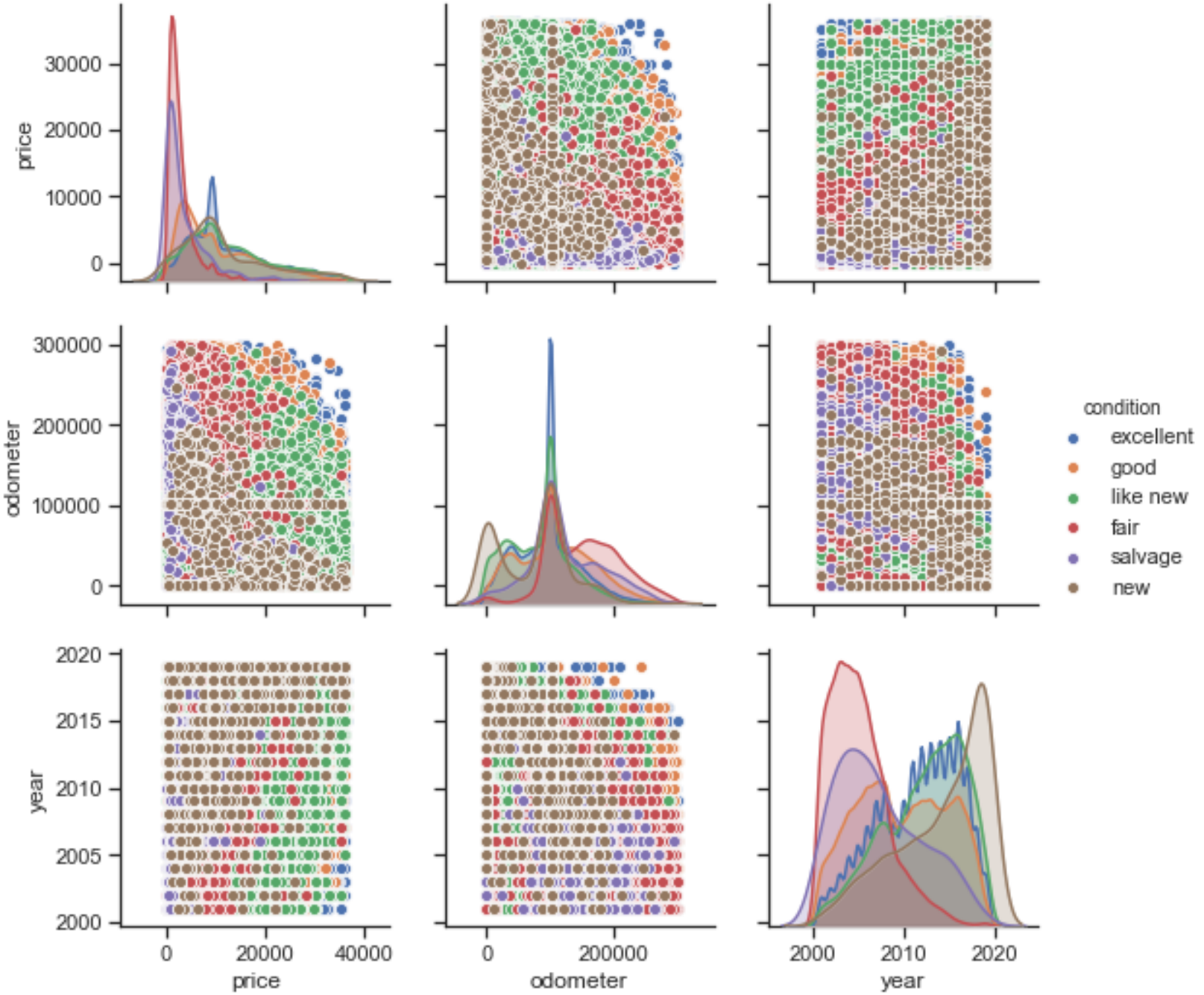
H0: Samples from different models(F150 and Escape) share the same mean price
p-value is 0.675

Fail to reject H0: There is no proof that samples from the two models share the same mean price.

Modeling Ideas



Modeling Ideas



Modeling

Dummy Regressor

```
from sklearn.dummy import DummyRegressor
from sklearn.metrics import r2_score
```

```
y= data_learn.price
X= data_learn.drop('price',axis=1)
```

```
dumr = DummyRegressor()
```

```
X_train_dumr, X_test_dumr,y_train_dumr, y_test_dumr=train_test_split(X,y,train_size=0.5, random_state=42)
```

```
dumr.fit(X_train_dumr, y_train_dumr)
y_predict_dumr=dumr.predict(X_test_dumr)
```

```
#calculate the RMSE of Training set
test_rmse_dumr = np.sqrt(MSE(y_test_dumr, y_predict_dumr))

print("Dummy Regression RMSE = {:.2f}".format(test_rmse_dumr))
train_rmse_dumr = np.sqrt(MSE(y_train_dumr, y_predict_dumr[1:]))
print("Dummy Regression RMSE = {:.2f}".format(train_rmse_dumr))
```

```
Dummy Regression RMSE = 8235.70
Dummy Regression RMSE = 8223.75
```

Linear Regression

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

```
y= data_learn.price
X= data_learn.drop('price',axis=1)
```

```
X_train_lr, X_test_lr,y_train_lr, y_test_lr=train_test_split(X,y,train_size=0.5, random_state=42)
```

```
lingr = LinearRegression()
```

```
lingr.fit(X_train_lr, y_train_lr)
y_predict_lr=lingr.predict(X_test_lr)
```

```
#calculate the RMSE of Training set
test_rmse_lr = np.sqrt(MSE(y_test_lr, y_predict_lr))

print("linear Regression RMSE = {:.2f}".format(test_rmse_lr))
train_rmse_lr = np.sqrt(MSE(y_train_lr, y_predict_lr[1:]))
print("linear Regression RMSE = {:.2f}".format(train_rmse_lr))
```

```
linear Regression RMSE = 6837.72
linear Regression RMSE = 9400.13
```

Random Forest Regressor

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble.forest import RandomForestClassifier
from sklearn.feature_selection import SelectFromModel
```

```
#sample for random Forest Feature S
data_learn_sample=data_learn.sample(n=2000)
```

```
y2= data_learn_sample.price
X2= data_learn_sample.drop('price',axis=1)
```

```
X_train_rf, X_test_rf,y_train_rf, y_test_rf=train_test_split(X2,y2,train_size=0.5, random_state=42)
```

```
sel = SelectFromModel(RandomForestClassifier(n_estimators=100, n_jobs=1))
sel.fit(X_train_rf, y_train_rf)
```

```
SelectFromModel(estimator=RandomForestClassifier(bootstrap=True,
                                                 class_weight=None,
                                                 criterion='gini',
                                                 max_depth=None,
                                                 max_features='auto',
                                                 max_leaf_nodes=None,
                                                 min_impurity_decrease=0.0,
                                                 min_impurity_split=None,
                                                 min_samples_leaf=1,
                                                 min_samples_split=2,
                                                 min_weight_fraction_leaf=0.0,
                                                 n_estimators=100, n_jobs=1,
                                                 oob_score=False,
                                                 random_state=None, verbose=0,
                                                 warm_start=False),
                max_features=None, norm_order=1, prefit=False, threshold=None)
```

```
sel.get_support()
```

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

```
y= data_learn.price
X= data_learn.drop('price',axis=1)
```

```
regressor = RandomForestRegressor()
```

```
X_train_rf, X_test_rf,y_train_rf, y_test_rf=train_test_split(X,y,train_size=0.5, random_state=42)
```

```
regressor.fit(X_train_rf, y_train_rf)
```

```
C:\Users\Administrator\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:245: FutureWarning: Th
nge from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=10,
                      n_jobs=None, oob_score=False, random_state=None,
                      verbose=0, warm_start=False)
```

```
y_predict_rf = lingr.predict(X_test_rf)
```

```
#calculate the RMSE of Testing set
test_rmse_rf= np.sqrt(MSE(y_test_rf , y_predict_rf))
print("Random Forrest RMSE= {:.2f}".format(test_rmse_rf))
train_rmse_rf = np.sqrt(MSE(y_train_rf, y_predict_rf[1:]))
print("Random Forrest RMSE = {:.2f}".format(train_rmse_rf))
```

```
Random Forrest RMSE= 6837.72
Random Forrest RMSE = 9400.13
```

GradientBoostingRegressor

```
from sklearn.ensemble import GradientBoostingRegressor
```

```
X_train_gbr, X_test_gbr,y_train_gbr, y_test_gbr=train_test_split(X,y,train_size=0.5, random_state=42)
```

```
grabr=GradientBoostingRegressor()
```

```
grabr.fit(X_train_gbr, y_train_gbr)
```

```
GradientBoostingRegressor(alpha=0.9, criterion='friedman_mse', init=None,
                          learning_rate=0.1, loss='ls', max_depth=3,
                          max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, n_estimators=100,
                          n_iter_no_change=None, presort='auto',
                          random_state=None, subsample=1.0, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False)
```

```
y_predict_gbr = grabr.predict(X_test_gbr)
```

```
#calculate the RMSE of Testing set
test_rmse_gbr= np.sqrt(MSE(y_test_gbr , y_predict_gbr))
print("gbr RMSE= {:.2f}".format(test_rmse_gbr))
train_rmse_gbr = np.sqrt(MSE(y_train_gbr, y_predict_gbr[1:]))
print("gbr RMSE = {:.2f}".format(train_rmse_gbr))
```

```
gbr RMSE= 5966.61
gbr RMSE = 9827.91
```

Modeling

KNN Regressor

```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import pairwise_distances
from sklearn import neighbors
from math import sqrt
from sklearn.metrics import mean_squared_error
```

```
X_train_knn, X_test_knn,y_train_knn, y_test_knn=train_test_split(X,y,train_size=0.2, random_state=42)
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))

X_train_scaled_knn = scaler.fit_transform(X_train_knn)
X_train_df_knn = pd.DataFrame(X_train_scaled_knn)

X_test_scaled_knn = scaler.fit_transform(X_test_knn)
X_test_df_knn = pd.DataFrame(X_test_scaled_knn )
```

```
rmse_val = []
for K in range(20):
    K += 1
    KNN = neighbors.KNeighborsRegressor(n_neighbors = K)

    KNN.fit(X_train_df_knn, y_train_knn)  #fit the model
    y_predict_knn=KNN.predict(X_test_df_knn) #make prediction on test set
    error = sqrt(mean_squared_error(y_test_knn, y_predict_knn)) #calculate rmse
    rmse_val.append(error) #store rmse values
    print('RMSE value for k= ' , K , 'is:', error )
```

```
RMSE value for k=  1 is: 6857.7385228997755
RMSE value for k=  2 is: 6285.13820044394
RMSE value for k=  3 is: 6132.749133829311
RMSE value for k=  4 is: 6066.297950887851
RMSE value for k=  5 is: 6029.790395103743
RMSE value for k=  6 is: 6016.762751297211
RMSE value for k=  7 is: 6004.324559883539
RMSE value for k=  8 is: 5993.122807271099
RMSE value for k=  9 is: 5988.8812481545365
RMSE value for k=  10 is: 5985.284740066363
RMSE value for k=  11 is: 5982.416496367859
RMSE value for k=  12 is: 5983.822922664758
RMSE value for k=  13 is: 5980.9763956382285
RMSE value for k=  14 is: 5977.312386299682
RMSE value for k=  15 is: 5978.9723233490595
RMSE value for k=  16 is: 5981.112865117809
RMSE value for k=  17 is: 5984.442828113159
RMSE value for k=  18 is: 5986.437700496936
RMSE value for k=  19 is: 5988.290390797117
RMSE value for k=  20 is: 5989.836284582579
```

```
X_train_knn, X_test_knn,y_train_knn, y_test_knn=train_test_split(X,y,train_size=0.5, random_state=42)
```

```
knn =neighbors.KNeighborsRegressor(n_neighbors = 3)
```

```
knn.fit(X_train_knn, y_train_knn)
```

```
KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                    weights='uniform')
```

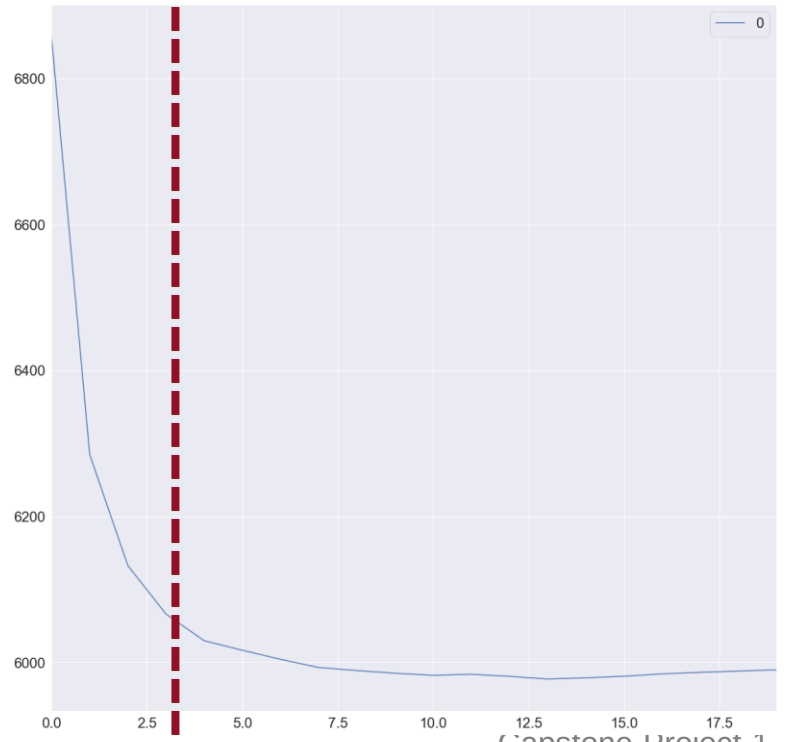
```
y_predict_knn = knn.predict(X_test_knn)
```

```
test_rmse_knn= np.sqrt(MSE(y_test_knn , y_predict_knn))
print("KNN RMSE= {:.2f}".format(test_rmse_knn))
train_rmse_knn = np.sqrt(MSE(y_train_knn, y_predict_knn[1:]))
print("KNN RMSE = {:.2f}".format(train_rmse_knn))
```

```
KNN RMSE= 6577.06
KNN RMSE = 10756.86
```

```
r2_test_knn=r2_score(y_test_knn, y_predict_knn)
r2_train_knn=r2_score(y_train_knn, y_predict_knn[1:])
```

K=3



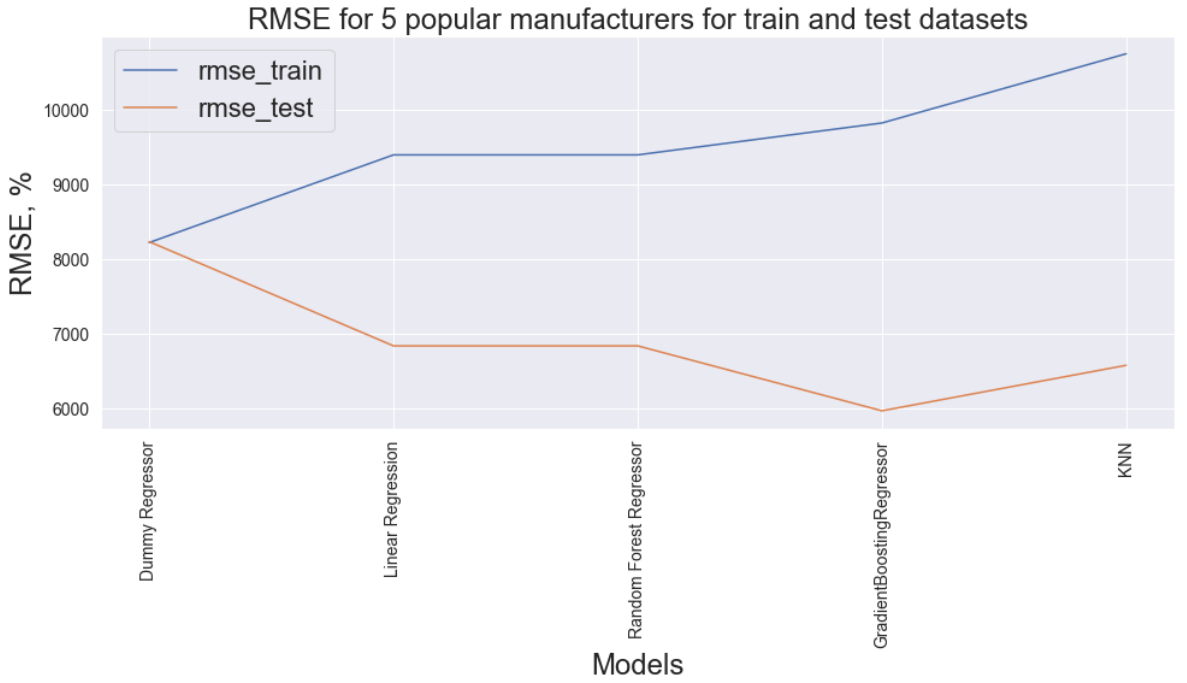
Compare Models

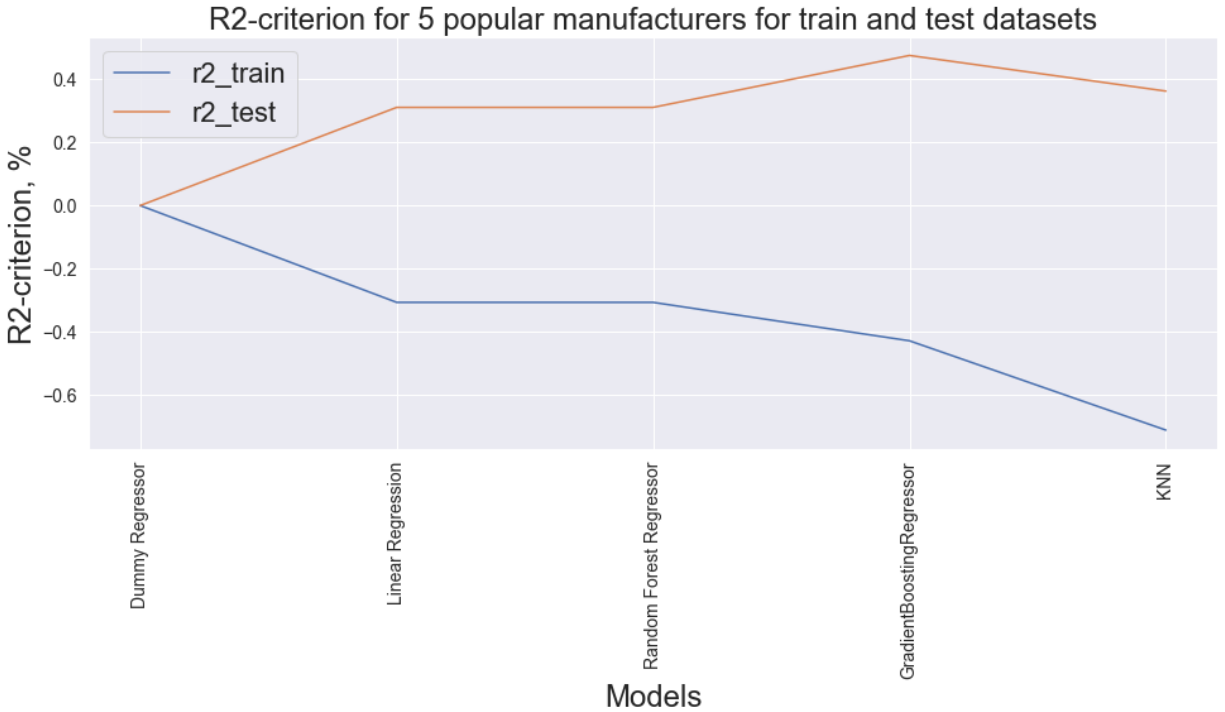
Out of 5 models, the best models by the RMSE is Gradient Boosting Regression.

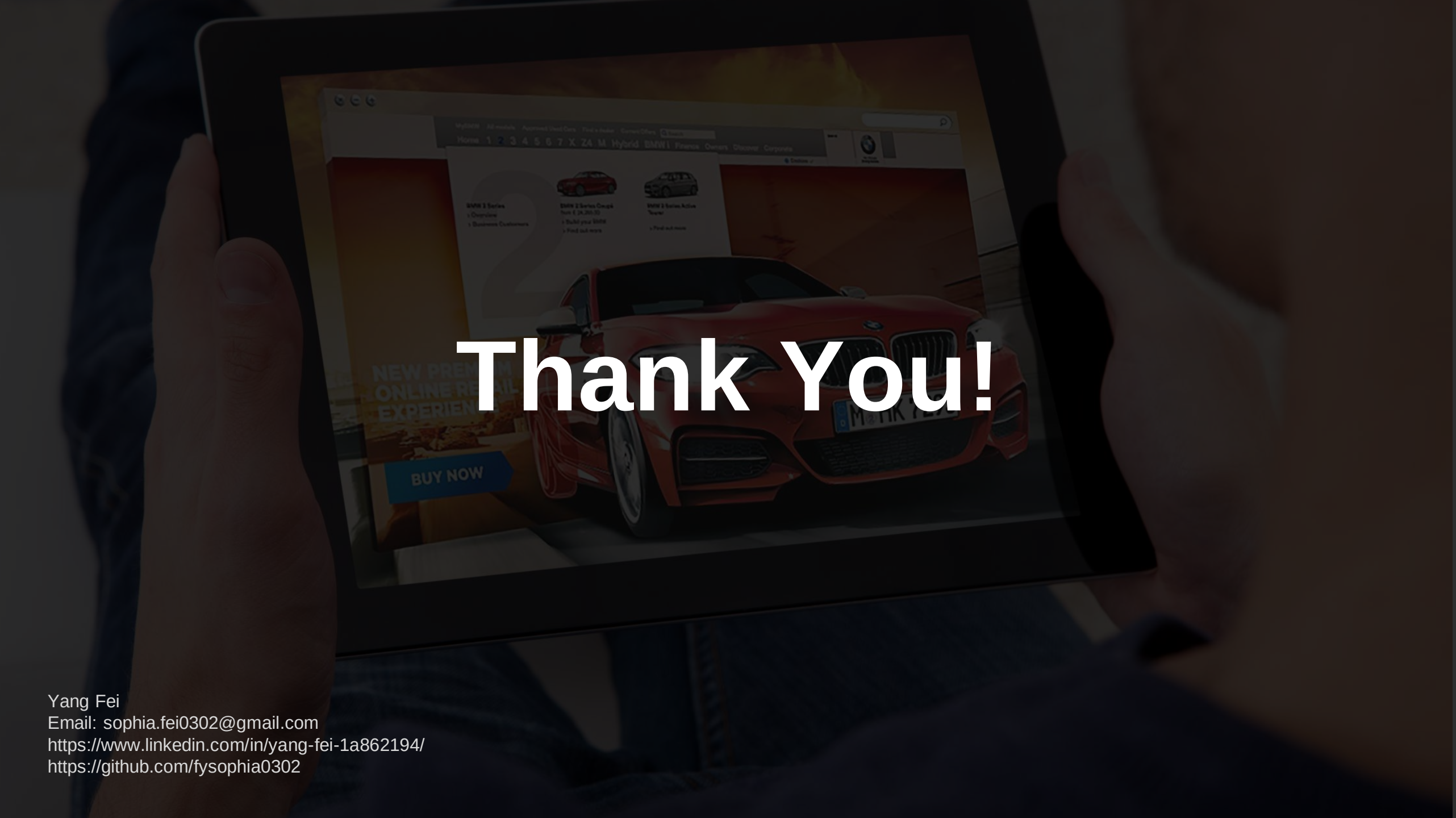
Determined by Random Forest modeling, price is more related with Manufacturers, Odometers, Condition, Paint Color.

Out of 24 features, we used only 7 features for the best model.

For KNN regression, the best K value should be 3.







Thank You!

Yang Fei
Email: sophia.fei0302@gmail.com
https://www.linkedin.com/in/yang-fei-1a862194/
https://github.com/fysophia0302