Predicting Used Car Resale Price

Daniel Kim

1. **Removing Outlier by Machine Learning Algorithm : Isolation Forest**

It is necessary to remove outliers effectively and efficiently in order to predict on the basis of machine learning algorithms. Judging outliers by limited numbers of features can lead to a big bias which may result in wrong prediction. So in order to run Isolation Forest, the dataset was processed with One Hot Encoding, and then it was processed with the Isolation Forest algorithm as seen below.

```
%time
from sklearn.ensemble import IsolationForest
clf=IsolationForest(n_estimators=50, max_samples=50, contaminatio
                    max_features=1.0, bootstrap=False, n_jobs=-1,
                    random_state=42, verbose=0,behaviour="new")
# 50 n_nodes, max 50 samples
# 0.04% outlier.
clf.fit(df)
pred = clf.predict(df)
df['anomaly']=pred
outliers=df.loc[df['anomaly']==-1]
outlier_index=list(outliers.index)
#print(outlier_index)
#Find the number of anomalies and normal points here points classified -
print(df['anomaly'].value_counts())
```

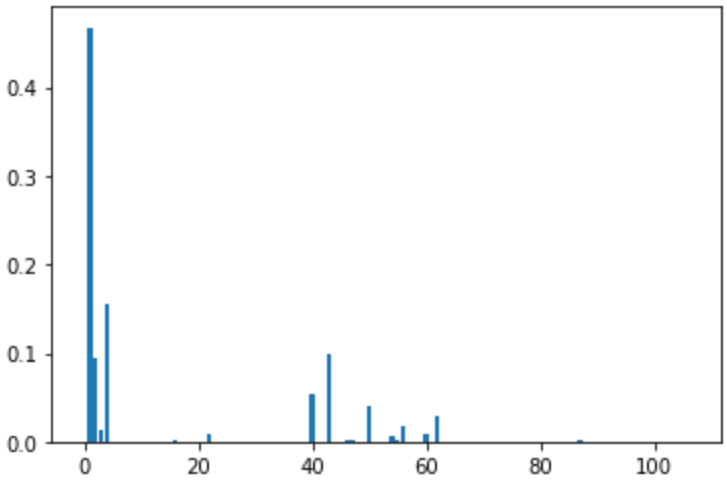
```
Wall time: 0 ns

C:\Users\danie\Anaconda\lib\site-packages\sklearn\ensemble\_iforest.py:255: FutureWar
ning: 'behaviour' is deprecated in 0.22 and will be removed in 0.24. You should not p
ass or set this parameter.
  FutureWarning

 1    517235
-1      2078
Name: anomaly, dtype: int64
```

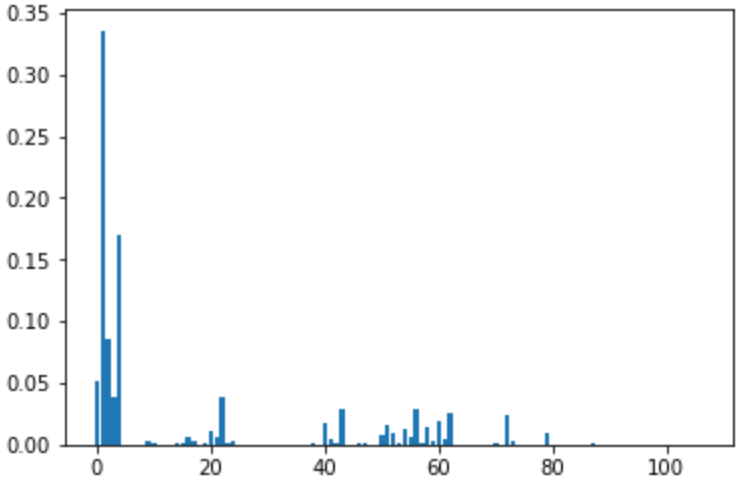
2. **Comparison of Feature Importance:**
   Decision Tree Regressor vs. Random Forest Regressor vs. Gradient Boost Regressor

   The below compares each algorithms' feature importances whose importance has more than 5% only.



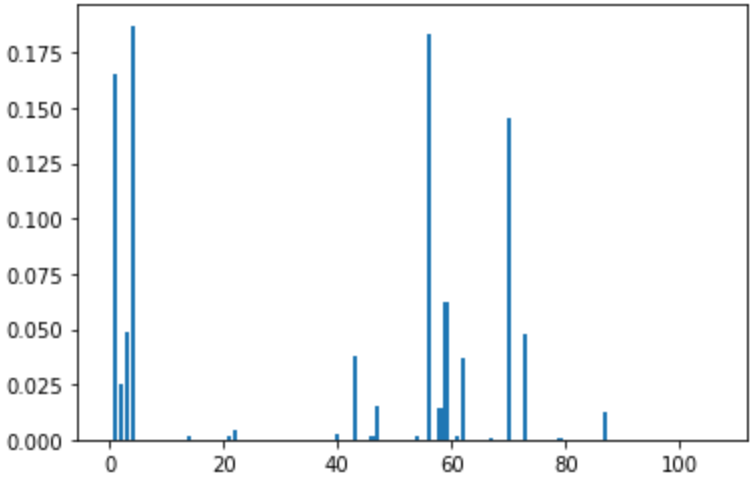
a. Decision Tree Regressor (x>=5% only)

Feature: 'odometer', Score: 0.46751
Feature: 'weather', Score: 0.09520
Feature: 'usage_yr', Score: 0.15504
Feature: 'Vclass_sport utility vehicle - 4wd', Score: 0.05271
Feature: 'VClass_standard pickup trucks 4wd', Score: 0.09778



b. Random Forest Regressor (x>=5% only)

Feature: 'cylinders', Score: 0.05090
Feature: 'odometer', Score: 0.33581
Feature: 'weather', Score: 0.08572
Feature: 'usage_yr', Score: 0.16935



c. Gradient Boost Regressor (x>=5% only)

Feature: 'odometer', Score: 0.16573
Feature: 'usage_yr', Score: 0.18730
Feature: 'division_Middle Atlantic', Score: 0.18348
Feature: 'manufacturer_cadillac', Score: 0.06249
Feature: 'manufacturer_chevrolet', Score: 0.14577

By comparing each feature importance from three different machine learning algorithms, 'odometer' and ' usage_yr' showed steady strong features when it comes to predicting 'price'. So we can conclude that 'odometer' and 'usage_yr' play key roles in predicting 'price'.

Interestingly, Gradient Boost Regressor showed such other strong features with more than 10% importance rates as 'division_Middle Atlantic' and ‘manufacturer_chevrolet’, which are One Hot Encoded for machine learning processes. This can be goot bridge to analyse further to identify how the target, ‘price’ and features are correlated to each other on such following conditions exclusively:: (1)‘division_Middle Atlantic' only, or (2) ‘manufacturer_chevrolet’, or both (1) and (2) altogether.

3. Comparison of Accuracy

Algorithms	Decision Tree Regressor	**Random Forest Regressor**	Gradient Boost Regressor
Accuracy (R-Squared)	-0.00018250127492003276	**0.005147258331855031**	-0.30837667790585555
Run Time	2.5 seconds	**853.5 minutes**	112.4 minutes
Best Parameters	'splitter': 'random', 'min_samples_split': 15, 'min_samples_leaf': 10, 'max_features': 'log2', 'max_depth': 5, 'criterion': 'friedman_mse'	**'n_estimators': 120, 'min_samples_split': 15, 'min_samples_leaf': 5, 'max_features': 'sqrt', 'max_depth': 15, 'criterion': 'mse'**	'n_estimators': 300, 'min_samples_split': 100, 'min_samples_leaf': 5, 'max_features': 'sqrt', 'max_depth': 25, 'loss': 'ls', 'criterion': 'friedman_mse'

Given the above test results, Random Forest Regressor showed best score (only positive as 0.005) in spite of enormous amount of Run Time (853.5 minutes). Gradient Boost Regressor showed disappointing result: the lowest score (-0.31) in consideration with a lot of Run Time taken.