Feature Importance

While training any machine learning model, we provide the algorithm with a bunch of features and corresponding label. Result obtained after the training holds the effect of all the features combined. However, it is not necessary that all of the features contribute equally in the prediction. Some feature might have more contribution in the prediction than the rest. It is also possible that some features are degrading the performance of the model and should be removed.

Analysis of features to know, how much each feature contributes in the prediction of model is done by feture importance. Through feature importance we also achieve interpretability of the model. If a feature contributes more in the prediction then that feature is probably important for the business. For example, while predicting the price of the house using features like number of bathrooms, bedrooms, locality etc, we need to know which feature contributes the most in the price of the house.

If we are dealing with a regression type of analysis, we can look at the coefficient of each feature to interpret feature importance. If the coefficient of a feature is large then it is probably an important feature. However, there are other popular techniques also that are used while analyzing feature importance. Some of them are listed below

Techniques to implement feature importance

a. Mean decrease impurity
b. Permutation Importance
c. Drop-Column Importance

Mean decrease impurity

This technique is used by scikit learn's RandomForestClassifier and RandomForestRegressor. This technique is defined as total decrease in node impurity (weighted by the probability of reaching that node (which is approximated by the proportion of samples reaching that node)) averaged over all trees of the ensemble.

Issue with mean decrease impurity -

Eventhough this technique is default feature importance technique in sklearn's RandomForest, it doesnot always give reliable results. This technique tends to inflate the importance of continuous or high-cardinality categorical variables. To demonstrate this, following experiment is performed on Kaggle's Two Sigma Connect: Rental Listing Inquiries-

In [1]:
```
from feature import *
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_boston
from sklearn.ensemble import RandomForestRegressor,RandomForestClassifier,GradientBoostingRegressor
from sklearn.metrics import *
import warnings
warnings.filterwarnings('ignore')
```

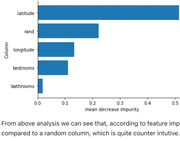
In [2]:
```
data = pd.read_csv('rent.csv')

columns = list(data.columns)
```

In this experiment we will try to add one column with random numbers in the data set and try to analyze the feature importance using mean decrease impurity. We will use RandomForestRegressor to fit the data

RandomForestRegressor

In [3]:
```
X = data[['bathrooms', 'bedrooms', 'longitude', 'latitude']]
y = data['price']
```

In [4]:
```
random = np.random.uniform(low=0.0,high=1000.0,size = len(X))
X['rand'] = random

rf = RandomForestRegressor(n_estimators=100,
         min_samples_leaf=1,
         n_jobs=-1,
         random_state=42)
rf.fit(X,y)

plot(dict(zip(X.columns,rf.feature_importances_)),'Mean decrease impurity feature importance','mean decrease im
```



From above analysis we can see that, according to feature importance, all columns except latitude are performing poorly as compared to a random column, which is quite counter intuitive. Therefore, we can say that, something is wrong with this approach.

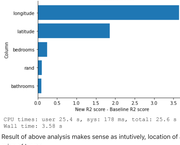
To solve this issue, other techniques like Permutation Importance and Drop-Column Importance can be used

Permutation Importance

In this technique first a model is trained with the existing data and a baseline metric is calcuated. Then each column is permuted turn by turn and new metric is calculated with the permuted data. If the difference between the new metric and baseline metric is large, then it suggests that permuted column was important. In this technique model is trained just once and metric from the permuted data is calulated using initially trained model.

Experiment performed above is again performed below, but instead of using mean decrease impurity, permutation importance is used

In [5]:
```
%%time
rf = RandomForestRegressor(n_estimators=100,
         min_samples_leaf=1,
         n_jobs=-1)
rf.fit(X,y)

imp = permutation_importances(rf,X,y,r2_score)

plot(imp,'Permutation Plot','New R2 score - Baseline R2 score')
```




```
CPU times: user 25.4 s, sys: 178 ms, total: 25.6 s
Wall time: 3.58 s
```

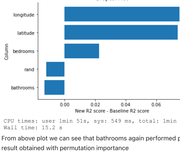
Result of above analysis makes sense as intutively, location of a house and number of bedrooms do plays an important role in the price of house

Drop-Column Importance

In this technique also we initially train a model with the whole data to calculate the baseline metric. But, instead of permuting the column, we drop the column turn by turn and retrain the model to calculate the new metric. Large difference in new metric and baseline metric indicates the dropped column was important. Contrary to the Permutation Importance, Drop-Column importance requires retraining of the model with each drop of the column

Same experiment is performed below by utilizing Drop-Column Importance

In [6]:
```
%%time
rf = RandomForestRegressor(n_estimators = 100,
         min_samples_leaf=1,
         n_jobs=-1)
rf.fit(X,y)

imp = dropcol_importances(rf,X,y,r2_score)

plot(imp,'Dropcol Plot','New R2 score - Baseline R2 score')
```




```
CPU times: user 1min 51s, sys: 549 ms, total: 1min 51s
Wall time: 15.2 s
```

From above plot we can see that bathrooms again performed poorly as compared to random column. This result is quite similar to the result obtained with permutation importance

Perfomances comparison

In the above results we can also see that drop column importance takes quite a long time to execute as compared to permutation imporatnce. This is because drop column importance technique retrains everytime it tries to calculate importance of a column, which is quite expensive. Whereas, permutation importance needs to be trained just once.

Time taken by Permutaion Importance - 214 ms

Time taken by Drop Column Importance - 616 ms

Effect of collinear features on importance

Lets analyse how a collinear features will affect permutation importance and drop-column importance

In this experiment we introduce a new column which is same as longitude column. This new column has complete correlation with original column and hence we can analyze its effect by placing noisy column and original column together

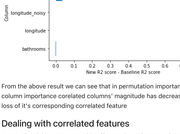
In [7]:
```
X.drop('rand',axis=1,inplace=True)
X['longitude_noisy'] = X.longitude
```

Permutation Importance

In [8]:
```
rf = RandomForestRegressor(n_estimators=100,
         min_samples_leaf=1,
         n_jobs=-1)
rf.fit(X,y)

imp = permutation_importances(rf,X,y,r2_score)

plot(imp,'Permutation Plot','New R2 score - Baseline R2 score')
```

Drop-Column Importance

In [9]:
```
rf = RandomForestRegressor(n_estimators=100,
         min_samples_leaf=1,
         n_jobs=-1)
rf.fit(X,y)

imp = dropcol_importances(rf,X,y,r2_score)

plot(imp,'Dropcol Plot','New R2 score - Baseline R2 score')
```



From the above result we can see that in permutation importance both correlated columns are shown important, Whereas in drop column importance correlated columns' magnitude has decreased because on dropping one column other column compensates the loss of it's corresponding correlated feature

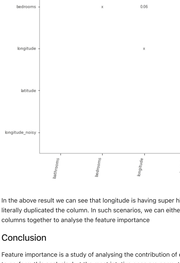
Dealing with correlated features

In a dataset it is necessary to identify correlated features. Otherwise, the results of feature importance techniques can become quite confusing. For example, eventhough in above experiment longitude importance is showing to be zero, we can not say that longitude is not important in predicting price. Reason for such result is presence of another correlated feature in the data.

To identify correalted feature we can use rfpimp package. Correlation matrix obtained from this package is upper triangular as a correlation matrix is always symmetrical across the diagonal hence lower triangular is redundant. Diagonal in this matrix is not present as correalation of a feature with itself is not important as it is always 1.

In [10]:
```
from rfpimp import plot_corr_heatmap
```

In [11]:
```
mat = plot_corr_heatmap(X,figsize=(10,8))
mat.view()
```



In the above result we can see that longitude is having super high correallion with longitude_noisy. It is as expected because we have literally duplicated the column. In such scenarios, we can either drop one of the columns, or permute all the pairs of of correalated columns together to analyse the feature importance

Conclusion

Feature importance is a study of analysing the contribution of each feature in a well trained model. Various techniques are employed to perform this analysis, but the most intutive ones are permutation and drop-column importance where we compare the performance of a baseline model (model with initial data) with the model that is trained after destroying the relationship of a column with the rest of data. In permutation importance we do this by permuting the column, whereas in drop column importance we do this by dropping the column and retraining the model.

While doing such analysis we should be vary of the fact that a dataset might also contain some correlated features. Hence they needs to be identified and then we can either drop one of the column or combine seperate pairs of correlated columns (drop/permute them together) while performing the analysis.