Class: (602) Introduction to Machine Learning

Professor: Dr. Ozgur Ozturk

Author for 'Homework 1 Section': Mr. Levan Sulimanov

Homework 1 section

After running the cells, add code to this notebook to achieve the following:

1) Split the data for training and testing, to use 80 percent as training data.

use 21 as your randomization seed (so you achieve same results for us to grade).

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html)

random_state = 21

train_size = .8

In the following steps use training set to fit the model, and test set to evaluate it.

In [256]:

```python
from sklearn.model_selection import train_test_split
import pandas as pd
from sklearn.metrics import mean_squared_error
import operator
from itertools import combinations
from sklearn.neighbors import KNeighborsRegressor
```

In [257]:

```python
print(X.shape)
print(y.shape)
```

```
(1460, 221)
(1460,)
```

In [258]:

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, random_
```

In [259]:

```
print("X_train.shape:", X_train.shape)
print("X_test.shape:", X_test.shape)
print("")
print("y_train.shape:", y_train.shape)
print("y_test.shape:", y_test.shape)
```

```
X_train.shape: (1168, 221)
X_test.shape: (292, 221)

y_train.shape: (1168,)
y_test.shape: (292,)
```

2) Analyze which feature alone would give the best prediction, list the scores and RMSE errors achieved by the top 10 predictors by score.

In [260]:

```
print("Number of features:", len(housing_ml.columns))
```

```
Number of features: 221
```

In [261]:

```
X_train[:, 0].reshape(-1, 1).shape
```

Out[261]:

```
(1168, 1)
```

In [262]:

```
feature_lst = list(housing_ml.columns)
feature_dict = {}

for f in range(0, len(feature_lst)):

    # training on single feature
    single_feat_train_data = X_train[:, f].reshape(-1, 1)
    single_feat_test_data = X_test[:, f].reshape(-1, 1)

    curr_linear_model = LinearRegression()
    curr_linear_model.fit(single_feat_train_data, y_train)
    curr_y_pred = curr_linear_model.predict(single_feat_test_data)

    curr_score = curr_linear_model.score(single_feat_test_data, y_test)
    curr_rmse = mean_squared_error(y_test, curr_y_pred, squared=False)
    # print("{} Feature's Score: {}".format(feature_lst[f], curr_score))
    # print("{} RMSE: {}".format(feature_lst[f], curr_rmse))
    # print("")
    col_indx = f

    feature_dict[f] = [feature_lst[f], curr_score, curr_rmse, col_indx]
```

In [263]:

```
sorted_by_score_top_10 = sorted(feature_dict.values(), key=operator.itemgetter(1
```

In [264]:

```
for best in sorted_by_score_top_10:
    print("{} [with indx={}] Feature's RMSE = {} with Score = {}".format(best[0],
    print("")
```

```
OverallQual [with indx=4] Feature's RMSE = 49018.44 with Score = 0.65

ExterQual_Coded [with indx=36] Feature's RMSE = 58260.15 with Score = 0.5

GrLivArea [with indx=16] Feature's RMSE = 61369.56 with Score = 0.44

KitchenQual_Coded [with indx=44] Feature's RMSE = 61622.3 with Score = 0.44

TotalBsmtSF [with indx=12] Feature's RMSE = 62430.33 with Score = 0.42

1stFlrSF [with indx=13] Feature's RMSE = 62876.26 with Score = 0.42

GarageCars [with indx=25] Feature's RMSE = 63131.02 with Score = 0.41

GarageArea [with indx=26] Feature's RMSE = 63377.71 with Score = 0.41

BsmtQual_Coded [with indx=38] Feature's RMSE = 66831.52 with Score = 0.34

GarageFinish_Coded [with indx=47] Feature's RMSE = 68071.65 with Score = 0.32
```

3) Select all possible 2 pairs of these top 10 predictors, and train 45 linear models, list the scores and RMSE errors achieved by the top 10 predictors by score.

In [265]:

```
get_col_indx = []
for idx in sorted_by_score_top_10:
    get_col_indx.append(idx[3])
print("Column indexes based on top 10 performing models:", get_col_indx)
```

```
Column indexes based on top 10 performing models: [4, 36, 16, 44, 12, 13, 25, 2
6, 38, 47]
```

In [266]:
```
# get all pair combinations from this list:
best_pairs = list(combinations(get_col_indx, 2))
print("Best pairs:", best_pairs)
print("")
print("Number of best pairs:", len(best_pairs))
```

```
Best pairs: [(4, 36), (4, 16), (4, 44), (4, 12), (4, 13), (4, 25), (4, 26), (4,
38), (4, 47), (36, 16), (36, 44), (36, 12), (36, 13), (36, 25), (36, 26), (36,
38), (36, 47), (16, 44), (16, 12), (16, 13), (16, 25), (16, 26), (16, 38), (16,
47), (44, 12), (44, 13), (44, 25), (44, 26), (44, 38), (44, 47), (12, 13), (12,
25), (12, 26), (12, 38), (12, 47), (13, 25), (13, 26), (13, 38), (13, 47), (25,
26), (25, 38), (25, 47), (26, 38), (26, 47), (38, 47)]

Number of best pairs: 45
```

In [267]:
```
X_train[:, (4,36)].shape
```

Out[267]:
```
(1168, 2)
```

In [268]:
```
housing_ml.columns[[0,2]]
```

Out[268]:
```
Index(['Id', 'LotFrontage'], dtype='object')
```

In [269]:
```
feature_lst = housing_ml.columns
feature_dict_best_pairs = {}


for curr_pair in best_pairs:

    pair = list(curr_pair)

    pair_feat_train_data = X_train[:, pair]
    pair_feat_test_data = X_test[:, pair]

    curr_pair_feat_linear_model = LinearRegression()
    curr_pair_feat_linear_model.fit(pair_feat_train_data, y_train)
    curr_pair_feat_y_pred = curr_pair_feat_linear_model.predict(pair_feat_test_da

    curr_pair_feat_score = curr_pair_feat_linear_model.score(pair_feat_test_data,
    curr_pair_feat_rmse = mean_squared_error(y_test, curr_pair_feat_y_pred, squar

    col_indx = "_and_".join(list(feature_lst[pair]))

    feature_dict_best_pairs[col_indx] = [col_indx, curr_pair_feat_score, curr_pai
```

In [270]:
```
sorted_by_score_pair_45_top_10 = sorted(feature_dict_best_pairs.values(), key=ope

for best in sorted_by_score_pair_45_top_10:
    print("{} [with indx={}] Feature's RMSE = {} with Score = {}".format(best[0],
```

```
OverallQual_and_1stFlrSF [with indx=[4, 13]] Feature's RMSE = 43764.69 with Sco
re = 0.72
OverallQual_and_TotalBsmtSF [with indx=[4, 12]] Feature's RMSE = 44577.66 with
Score = 0.71
OverallQual_and_GrLivArea [with indx=[4, 16]] Feature's RMSE = 45673.47 with Sc
ore = 0.69
OverallQual_and_GarageArea [with indx=[4, 26]] Feature's RMSE = 46468.13 with S
core = 0.68
OverallQual_and_KitchenQual_Coded [with indx=[4, 44]] Feature's RMSE = 47070.07
with Score = 0.67
OverallQual_and_GarageCars [with indx=[4, 25]] Feature's RMSE = 47124.91 with S
core = 0.67
OverallQual_and_ExterQual_Coded [with indx=[4, 36]] Feature's RMSE = 47433.82 w
ith Score = 0.67
OverallQual_and_BsmtQual_Coded [with indx=[4, 38]] Feature's RMSE = 48237.75 wi
th Score = 0.66
OverallQual_and_GarageFinish_Coded [with indx=[4, 47]] Feature's RMSE = 48571.2
7 with Score = 0.65
ExterQual_Coded_and_1stFlrSF [with indx=[36, 13]] Feature's RMSE = 49640.64 wit
h Score = 0.64
```

4) Train a single model using all features. Calculate RMSE and score. Observe how much of the prediction power was in the 2 pairs, vs all features.

In [271]:
```
all_feat_linear_model = LinearRegression()
all_feat_linear_model.fit(X_train, y_train)
all_feat_y_pred = all_feat_linear_model.predict(X_test)

all_feat_score = all_feat_linear_model.score(X_test, y_test)
all_feat_rmse = mean_squared_error(y_test, all_feat_y_pred, squared=False)

curr_all_feat_score = all_feat_linear_model.score(X_test, y_test)
curr_all_feat_rmse = mean_squared_error(y_test, all_feat_y_pred, squared=False)

print("All feature score:", curr_all_feat_score)
print("RMSE Score:", curr_all_feat_rmse)
```

```
All feature score: 0.8127547098528441
RMSE Score: 35623.30345000228
```

In [3]:
```
#---------------------------------------------------#
```

In [3]:
```
#---------------------------------------------------#
```

5) Use the 5NN and 10NN regressor with all features, and list the RMSE and score for these 2 models

[https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html (https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html)](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html)

In [272]:

```
# 5NN regressor
knn_5_model = KNeighborsRegressor(n_neighbors=5)
knn_5_model.fit(X_train, y_train)
y_pred_knn_5 = knn_5_model.predict(X_test)

knn_5_score = knn_5_model.score(X_test, y_test)
knn_5_rmse = mean_squared_error(y_test, y_pred_knn_5, squared=False)

print("KNN 5's Score:", knn_5_score)
print("KNN 5's RMSE:", knn_5_rmse)

# 10NN regressor
knn_10_model = KNeighborsRegressor(n_neighbors=10)
knn_10_model.fit(X_train, y_train)
y_pred_knn_10 = knn_10_model.predict(X_test)

knn_10_score = knn_10_model.score(X_test, y_test)
knn_10_rmse = mean_squared_error(y_test, y_pred_knn_10, squared=False)

print("KNN 5's Score:", knn_10_score)
print("KNN 5's RMSE:", knn_10_rmse)
```

```
KNN 5's Score: 0.6015421069563245
KNN 5's RMSE: 51966.07983368139
KNN 5's Score: 0.5949812006956909
KNN 5's RMSE: 52392.16317256475
```

observe if the results are better than linear regression?

In [273]:

```
# Answer: No they are worse by around ~20%
```

Which regressor is better for inference?

Top 3 models for inference:

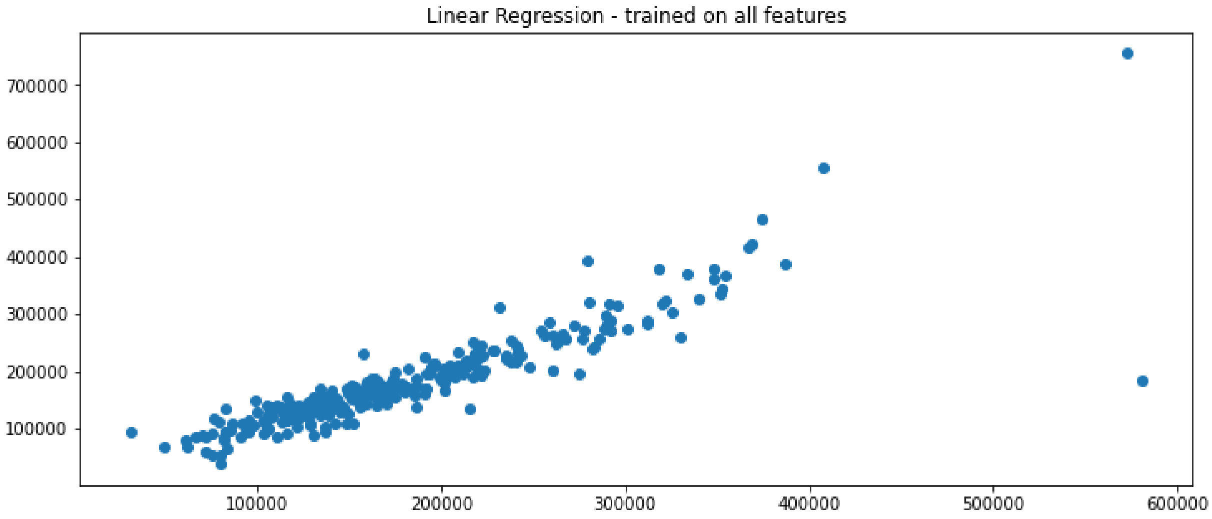
- 1. Linear Regression trained on all features
- 2. Linear Regression with 2 selected features [OverallQual, 1stFlrSF]
- 3. Linear Regression with 2 selected features [OverallQual, TotalBsmtSF]

Thus, Linear Regression is better for the current analysis. However, as we know, many parameters can be tweaked, so if more options were done further, we may get alternative answers. E.g. KNN with higher number of neighbors + PCA analysis on best features could produce better results than just training linear regression model on all features.

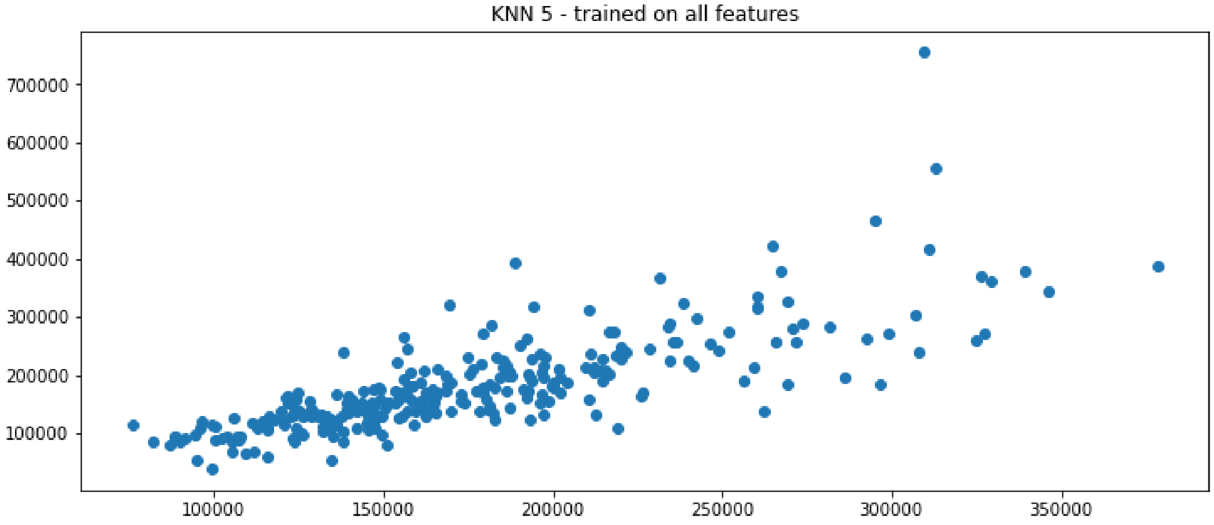
In [1]:
```
#----------------------------------------------------------#
```

In [2]:
```
# Plots for models of interest:
```

In [278]:
```
plt.figure(figsize=(12, 5))
plt.title("Linear Regression - trained on all features")
plt.scatter(all_feat_y_pred, y_test);
```



In [279]:
```
plt.figure(figsize=(12, 5))
plt.title("KNN 5 - trained on all features")
plt.scatter(y_pred_knn_5, y_test);
```


KNN 5 - trained on all features