

## House Prices - Advanced Regression Techniques

Σοφοκλής Κυριάκου

Στυλιανός Σοφοκλέους

Σταύρος Σπύρου

**Διδάσκοντες:**

Γιώργος Πάλλης

Πάυλος Αντωνίου.

## Περιγραφή Προβλήματος

Ο διαγωνισμός "House Prices: Advanced Regression Techniques" αφορά κατοικίες στο Ames της Αϊόβα και σκοπός είναι η δημιουργία ενός μοντέλου που μπορεί να προβλέψει με ακρίβεια την τιμή πώλησης κάθε σπίτι με βάση ένα σύνολο χαρακτηριστικών εισόδου.

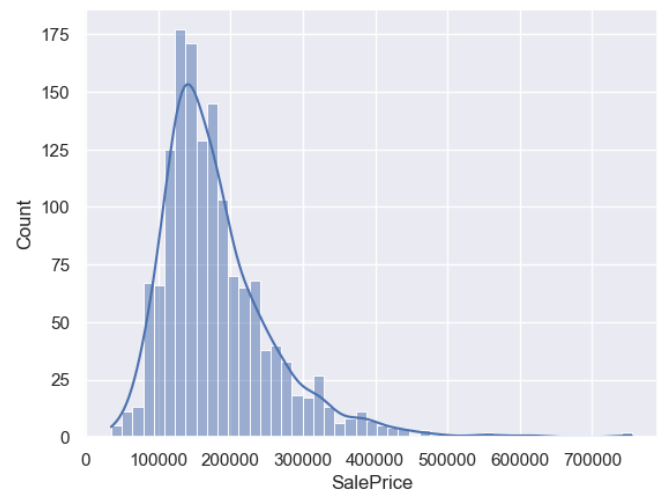
## Περιγραφή δεδομένων

Το σύνολο δεδομένων που παρέχεται για αυτόν τον διαγωνισμό περιέχει 81 χαρακτηριστικά των σπιτιών εκ των οποίων τα δύο είναι η τιμή πώλησης κάθε σπιτιού και το id του. Τα χαρακτηριστικά εισόδου είναι ένας συνδυασμός αριθμητικών και λεκτικών μεταβλητών, συμπεριλαμβανομένων χαρακτηριστικών όπως ο αριθμός των υπνοδωματίων και των μπάνιων, το μέγεθος του οικοπέδου, ο τύπος υλικού στέγης και η γειτονιά στην οποία βρίσκεται το σπίτι. Το σύνολο δεδομένων περιέχει συνολικά 1460

παραδείγματα, χωρισμένα σε ένα σετ εκπαίδευσης και σε ένα σύνολο δοκιμών επίσης 1460 παραδειγμάτων.

## Preprocessing

### Παρατήρηση Sale Price



Αρχικά για να αντιληφθούμε τα δεδομένα μας καλύτερα δημιουργήσαμε τα απαραίτητα γραφήματα για συγκεκριμένα key features και συγκεκριμένες μετρικές.

```
1 dataset['SalePrice'].describe()
[12] ✓ 0.0s
... count      1460.000000
    mean    180921.195890
    std     79442.502883
    min     34900.000000
    25%    129975.000000
    50%    163000.000000
    75%    214000.000000
    max     755000.000000
    Name: SalePrice, dtype: float64
```

Παρατηρώντας τα χαρακτηριστικά της τιμής πώλησης των σπιτιών βλέπουμε το εξής:

- 1)  $Q3 - Q2 = 214000 - 163000 = 51000$
- 2)  $Q2 - Q1 = 163000 - 129975 = 33025$

(1) > (2) Επιβεβαιώνετε ότι έχουμε positively skewed δεδομένα, και θα χρειαστεί οπωσδήποτε transformation σε πολλά από τα features.

## Ανάλυση άδειων στηλών

```
1 # %%
2 pd.set_option('display.max_rows', None)
3 dataset.isnull().sum()
✓ 0.0s
```

Output exceeds the [size limit](#). Open the full output data [in a text editor](#).

Id	0
MSSubClass	0
MSZoning	0
LotFrontage	259
LotArea	0
Street	0
Alley	1369
LotShape	0
LandContour	0
Utilities	0
LotConfig	0
LandSlope	0
Neighborhood	0
Condition1	0
Condition2	0
BldgType	0
HouseStyle	0
OverallQual	0
OverallCond	0
YearBuilt	0
YearRemodAdd	0
RoofStyle	0
RoofMatl	0
Exterior1st	0
Exterior2nd	0
...	
YrSold	0
SaleType	0
SaleCondition	0
SalePrice	0

dtype: int64

```
1 # %%
2 pd.set_option('display.max_rows', None)
3 dataset[['LotFrontage', 'Alley', 'FireplaceQu', 'PoolQC',
4          'Fence', 'MiscFeature', 'GarageType', 'GarageFinish']].isnull().sum()
✓ 0.0s
```

LotFrontage	259
Alley	1369
FireplaceQu	690
PoolQC	1453
Fence	1179
MiscFeature	1486
GarageType	81
GarageFinish	81

dtype: int64

```
14 for i, column in enumerate(dataset):
15     if (dataset[column].dtype == object):
16         dataset[column] = dataset[column].fillna(dataset[column].mode()[0])
17     else:
18         dataset[column] = dataset[column].fillna(dataset[column].mean())
19
20
```

Χρησιμοποιώντας το πιο πάνω κώδικα παρατηρήσαμε ποιες στήλες έχουν πολλά κενά και πιθανότατα να πρέπει να τις κάνουμε drop από τα δεδομένα μας ή να προχωρήσουμε στο να γεμίσουμε εμείς τα συγκεκριμένα κενά με την χρήση του πιο συχνά εμφανιζόμενου στοιχείου στην περίπτωση που η στήλη ήταν string και με τον μέσο όρο στην περίπτωση που ήταν αριθμητικό το περιεχόμενο της στήλης. Οι στήλες που φαίνονται στην εικόνα έχουν γίνει drop.

## String/Categorical Features

Κάποια από τα δεδομένα μας δεν ήταν αριθμητικά. Τα δεδομένα αυτά έπρεπε να μορφοποιηθούν κατάλληλα σε αριθμητικά δεδομένα έτσι ώστε οι regressors να μπορούν να δουλέψουν πάνω στα δεδομένα αυτά. Εκλέξαμε 2 διαφορετικούς τρόπους μορφοποίησης, Label Encoding και Dummy Variables. Ενώ στη θεωρία τα dummy variables είναι καλύτερα, παρατηρήσαμε πως στην προκειμένη περίπτωση το Label Encoding έδωσε καλύτερα αποτελέσματα. Για να είμαστε σίγουροι πως τα δεδομένα του test set θα μορφοποιούνταν με τον ίδιο τρόπο όπως και αυτά του training set, χρησιμοποιήσαμε τον ίδιο label encoder για κάθε feature.

```
1 # Categorical Data Encoding
2 # Categorical features
3 cat_features = np.array([i for i in dataset.columns.tolist() if dataset[i].dtype == 'object'])
4 enc_list = {}
5
6 for i in cat_features:
7     enc_list[i] = preprocessing.LabelEncoder()
8     dataset[i] = enc_list[i].fit_transform(dataset[i])
✓ 0.0s
```

## Highly Correlated Features

Παρατηρώντας το correlation όλων των features μεταξύ τους παρατηρήσαμε ότι υπήρχαν κάποια που είχαν αρκετά ψηλό δείκτη correlation και θα έπρεπε κάποια από αυτά τα χαρακτηριστικά να γίνουν drop.

CarageCars -> GarageArea

**Επεξήγηση:** Τα δεδομένα για το garagcars είναι 1 ή 2 σε αντίθεση με το garage area το οποίο περιέχει το εμβαδό του χώρου. Οι 2 αυτές μετρικές είναι σχεδόν άμεσα συσχετιζόμενες (correlation = 0.88) αφού όσο μεγαλύτερος είναι ο χώρος τόσο περισσότερα αυτοκίνητα θα μπορούν να σταθμεύσουν

- Totrmsabvgrd, 1stFlrSf -> GrLivArea

**Επεξήγηση:** Παρατηρήσαμε πως η συσχέτιση των δύο στηλών Totrmsabvgrd,

1stFlrSf είχαν αρκετά υψηλή τιμή με το GrLivArea οπότε αποφασίσαμε να αφήσουμε το GrLivArea. Η συσχέτιση του Totrmsabvgrd και GrLivArea είναι ψηλή αφού όσο μεγαλύτερο το εμβαδόν του σπιτιού είναι λογικό να έχει περισσότερα δωμάτια.

- Exterior2nd -> Exterior1st

**Επεξήγηση:** Παρατηρήσαμε πως το Exterior2nd και Exterior1st έχουν ψηλή συσχέτιση και αποφασίσαμε να κρατήσουμε το Exterior1st.

Παρατηρήσαμε πως τα περισσότερα σπίτια έχουν μόνο μια επένδυση του ίδιου υλικού και γ' αυτό έχουν υψηλή συσχέτιση.

- GarageYrBlt (χρονολογία που κτίστηκε το garage) -> YearBUILT (χρονολογία που κτίστηκε το σπίτι)

**Επεξήγηση:** Τις περισσότερες φορές οι 2 χρονολογίες συμπίπτουν ή έχουν πολύ λίγη διαφορά σε χρόνια. Επιλέξαμε να κρατήσουμε την χρονολογία που κτίστηκε το σπίτι γιατί πιστεύαμε ότι είναι πολύ πιο σημαντική.

- MSSubClass -> BldgType

**Επεξήγηση:** Παρατηρήσαμε υψηλή συσχέτιση μεταξύ των δύο στηλών και αποφασίσαμε να κρατήσουμε το BldgType λόγο του ότι είχε λιγότερα attributes. Και οι δύο στήλες περιγράφουν τον τύπο του σπιτιού με την διαφορά ότι το MSSubClass έχει περισσότερη λεπτομέρεια στο είδος του σπιτιού.

## Low Correlated Features

Παρατηρώντας το correlation όλων των features σε σχέση με την τιμή πώλησης που είναι και το ζητούμενο στο συγκεκριμένο πρόβλημα παρατηρήσαμε ότι κάποια features είχαν πολύ χαμηλό correlation άρα δεν είχαν κάποια σχέση/εξάρτηση με την τιμή οπότε τα

συγκεκριμένα χαρακτηριστικά τα κάναμε drop.

Feature	Correlation with Sale price
Street	0.04
LandContour	0.02
Utilities	-0.01
LotConfig	-0.07
LandSlope	0.05
Condition2	0.01
MasVnrType	0.02
BsmtCond	0.12
BsmtFinType2	0.03
BsmtFinSF2	-0.01
BsmtHalfBath	-0.02
LowQualFinSf	-0.03
3SsnPorch	0.04
MiscVal	-0.01
MoSold	0.05
YrSold	-0.03
RoofMatl	0.13
Heating	-0.09
Electrical	0.23
Functional	0.11
GarageQual	0.09
GarageCond	0.13
PavedDrive	0.23
PoolArea	0.09

## Drop extreme Outliers



Οπτικά, έχουμε αφαιρέσει κάποιες τιμές οι οποίες ήταν extreme outliers στον κανόνα. Για παράδειγμα, βλέπουμε πως το LotArea περιέχει τιμές οι οποίες εμφανίζονται πολύ λίγες φορές και για αυτό τις αφαιρέσαμε.

LotArea skewness before removing outliers: 12.2

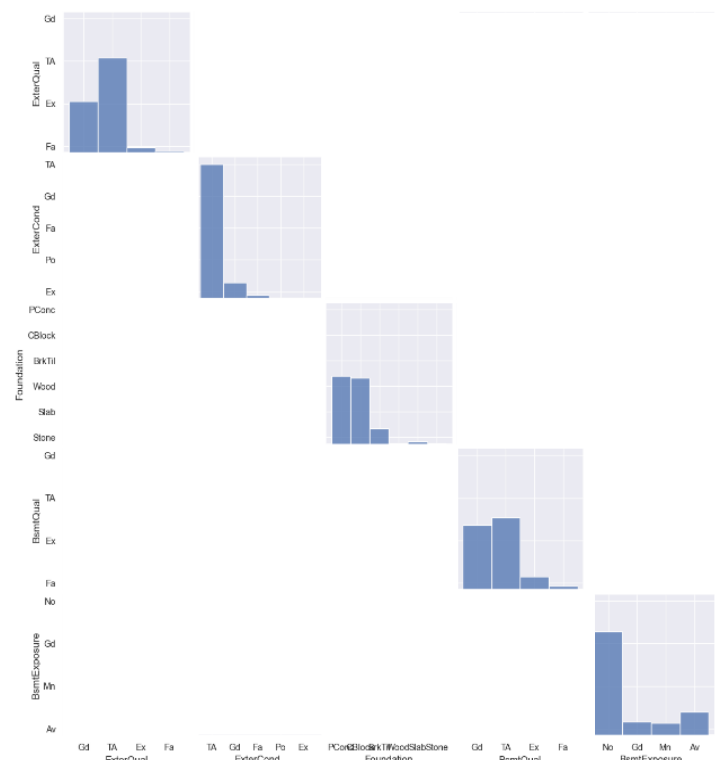
LotArea skewness before removing outliers: 2.2

Παρομοίως, έχουμε αφαιρέσει τιμές από άλλα features τα οποία παρουσίαζαν παρόμοια συμπεριφορά.

```
1 # Drop outliers
2 dataset = dataset[(dataset['LotArea'] < 50000)]
3 dataset = dataset[(dataset['MasVnrArea'] < 800)]
4 dataset = dataset[(dataset['BsmntFinSF1'] < 2300)]
5 dataset = dataset[(dataset['TotalBsmntSF'] < 5000)]
6 dataset = dataset[(dataset['GrLivArea'] < 4000)]
7 dataset = dataset[(dataset['OpenPorchSF'] < 380)]
8
```

## Drop Unbalanced Categorical Features

Κάποια από τα χαρακτηριστικά τα οποία αφορούσαν κατηγοριοποίηση ήταν σε μεγάλο βαθμό unbalanced. Σε πολλές περιπτώσεις μια από τις κατηγορίες είχε την πλειοψηφία των παρατηρήσεων και οι υπόλοιπες κατηγορίες είχαν ελάχιστες. Μετά από δοκιμές που κάναμε καταλήξαμε στο συμπέρασμα ότι αποβάλλοντας τις πιο κάτω στήλες που ήταν πολύ unbalanced είχαμε αισθητά καλύτερα αποτελέσματα. Στις πιο κάτω γραφικές παραστάσεις φαίνεται το πιο πάνω φαινόμενο.



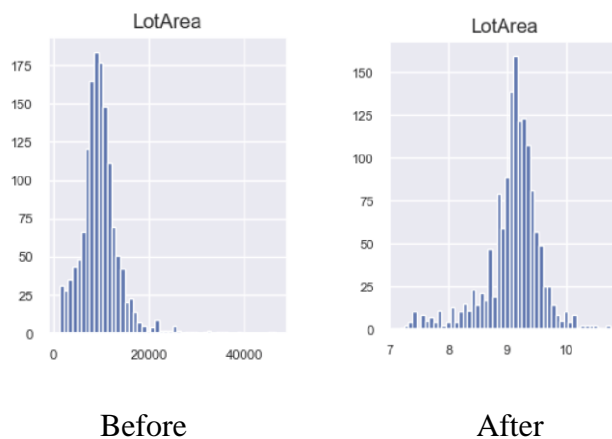
## Converting Porch Area to Binary

Τα χαρακτηριστικά EnclosedPorch και ScreenPorch αφορούσαν τα τετραγωνικά μέτρα συγκεκριμένων ειδών εσωτερικής αυλής τα οποία είχαν σχετικά πολύ χαμηλό correlation (<0.1). Στην προσπάθεια μας να βελτιστοποιήσουμε το τελικό αποτέλεσμα μας αν και αρχικά είχα κάνει drop τα συγκεκριμένα χαρακτηριστικά αποφασίσαμε να τα κάνουμε binary, δηλαδή αν κάποιο κτήριο είχε στις συγκεκριμένες τιμή μεγαλύτερη από το 0 τότε βάζαμε 1 αλλιώς βάζαμε 0. Αυτό αύξησε το correlation των 2 στηλών με το sale price στο περίπου 0.2. Επίσης βελτίωσε το τελικό error κατά 0.004.

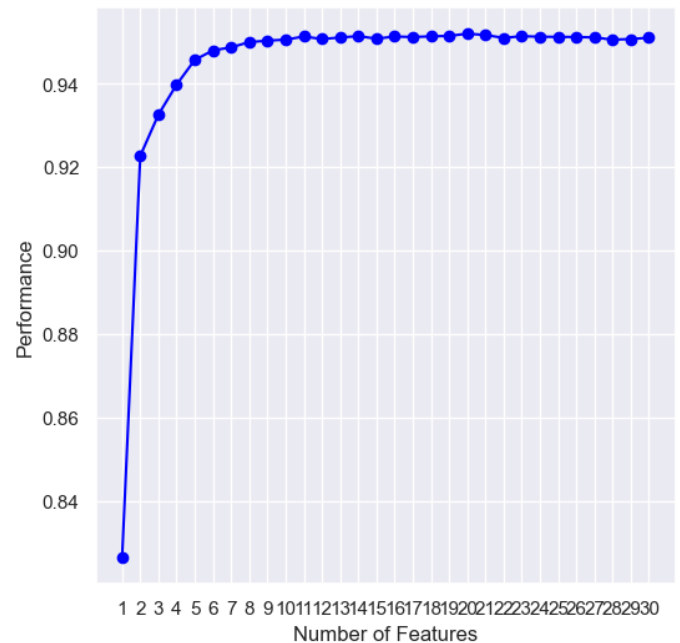
## Unskewing transformations

Προκειμένου να κάνουμε unskew τα δεδομένα μας, δοκιμάσαμε δυο τεχνικές για unskewing. Αρχικά δοκιμάσαμε να κάνουμε BoxCox transformation στα δεδομένα μας. Σύμφωνα με τα τελικά

αποτελέσματα που πήραμε μετά την παράδοση των τελικών αποτελεσμάτων στην πλατφόρμα ξανατρέξαμε τους αλγορίθμους αλλά με την χρήση του φυσικού λογάριθμου (ln). Ο φυσικός λογάριθμος είναι μια καλή τεχνική transformation αφού επιτυγχάνει να κανονικοποιησει τα δεδομένα. Η χρήση του ln φάνηκε να είναι καλύτερη παρά του BoxCox και γι' αυτό επιλέξαμε τον λογάριθμο.



περιέχει όλα τα features εκτός τις τιμές των σπιτιών και το Y το οποίο περιέχει τον στόχο, δηλαδή τις τιμές. Ο διαγωνισμός κρίνεται με το χαμηλότερο RMSE (Root mean square error). Δημιουργήσαμε την rmse συνάρτηση η οποία επιστρέφει το RMSE 2 τιμών για να χρησιμοποιηθεί από τον sequential selector. Ο forward selector τρέχει όλους τους δυνατούς συνδυασμούς από features και μας δίνει τον καλύτερο συνδυασμό.



## Feature Selection

```
1 from sklearn.ensemble import RandomForestRegressor
2 from sklearn.datasets import make_regression
3 from mlxtend.feature_selection import SequentialFeatureSelector as SFS
4 from mlxtend.plotting import plot SequentialFeatureSelection as plot_sfs
5 from sklearn.metrics import mean_squared_error
6 from sklearn.metrics import make_scorer
7
8 array = dataset.values
9 testarray = test.values
10 n = dataset.shape[1]
11 n = n-1
12
13 # test_X = testarray[:, 0:n] # features
14 X = dataset.iloc[:,0:n] # features
15 Y = dataset['SalePrice'] # target
16
17 ✓ 0.0s
18
19 # XX = dataset.iloc[:, ~dataset.columns.isin(['Id', 'SalePrice'])]
20 # YY = dataset['SalePrice']
21
22 def rmse(y_true, y_pred):
23     return 1 - np.sqrt(mean_squared_error(y_true, y_pred))
24
25 scorer = make_scorer(rmse, greater_is_better=True)
26
27 rfr = RandomForestRegressor()
28 sfs_range = SFS(estimator=rfr,
29                 k_features=(6,30),
30                 forward=True,
31                 floating=False,
32                 scoring=scorer,
33                 cv=5, n_jobs=8)
34
35 sfs_range.fit(X,Y)
36
37 # print the accuracy of the best combination as well as the set of best features
38 print('best combination (ACC: %.3f): %s\n' % (sfs_range.k_score_, sfs_range.k_feature_idx_))
39
40 plt.rcParams["figure.figsize"] = (6,6)
41 # use the plot_sfs to visualize all accuracies
42 plot_sfs(sfs_range.get_metric_dict(), kind='std_err')
43
44 ✓ 1m 50.4s
```

Για το feature selection δημιουργήσαμε δυο νέα data frames, το X το οποίο

Βλέπουμε πως ο forward selector έφτασε στο 95% accuracy

```
1 from sklearn.discriminant_analysis import StandardScaler
2 from sklearn.pipeline import Pipeline
3
4 pipeline6 = Pipeline([
5     ('scaler', StandardScaler()),
6     ('clf6', ensemble.GradientBoostingRegressor())
7 ])
8
9
10 param_grid6 = {
11     'clf6_n_estimators': [1000,2000,3000,4000],
12     'clf6_random_state': [1000,1234,1269],
13     'clf6_max_features': ['auto', 'sqrt', 'log2'],
14     'clf6_max_depth': [3,5, 15],
15     'clf6_min_samples_split': [2, 5,10,15],
16     'clf6_min_samples_leaf': [1, 3, 6,11,15],
17 }
18
19 grid_search6 = GridSearchCV(pipeline6, param_grid6, cv=5, n_jobs=8)
20 grid_search6.fit(X,Y)
21
22 print('Parameters : ', grid_search6.best_params_, '\nAccuracy Score : ', grid_search6.best_score_)
```

Στη συνέχεια, αφού πήραμε τα καλύτερα features από τον sequential selector, δημιουργήσαμε 6 pipelines με τα οποία δοκιμάσαμε συνδυασμούς από διάφορους








regressors και παραμέτρους τους για να βρούμε ποιος regressor θα μας δώσει το καλύτερο αποτέλεσμα. Καταλήξαμε στον Gradient Boosting Regressor ο οποίος μας έδωσε το χαμηλότερο RMSE ίσο με  $\sim 0,11$ .

```
1 print("GradientBoostingRegressor")
2 g_best = ensemble.GradientBoostingRegressor(n_estimators=1000, random_state=1234, learning_rate=0.02, max_depth=3,
3                                             max_features='log2', min_samples_leaf=11, min_samples_split=15, loss='huber').fit(x_train, y_train)
4 g_best_result = g_best.predict(x_test)
5 print('RMSE: {}'.format(np.sqrt(mean_squared_error(g_best_result, y_test))))
6 print('RMSE: {}'.format(np.sqrt(mean_squared_error(g_best_result, y_test))))
7 scores = cross_val_score(g_best, X, y, cv=5)
8 print('Accuracy: %.2f (%.2f - %.2f)' % (scores.mean(), scores.std() * 2))
```

Τέλος, χρησιμοποιήσαμε τον gradient boosting regressor με τις παραμέτρους που πήραμε από το pipeline του gradient boosting regressor και τρέξαμε το test αρχείο με αυτό τον regressor.

```
1 result = np.exp(g_best.predict(test.iloc[:, [ ... ]]))
2 output = pd.DataFrame({'Id': test.Id, 'SalePrice': result})
3 print(output)
4 output.to_csv('submission.csv', index=False)
```

Καταλήξαμε με score στο Kaggle 0.12402 στη θέση 563.

563	skylla wallo		0.12402	26	1d
Your Best Entry! Your submission scored 0.12402, which is an improvement of your previous score. Keep trying!					
564	hustle		0.12403	1	3d
595	HULT_SF_70	   	0.12409	19	2mo
566	Yusuf Dny		0.12411	34	20d