```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
%matplotlib inline
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt  # Matlab-style plotting
import seaborn as sns
color = sns.color_palette()
sns.set_style('darkgrid')
from scipy import stats
from scipy.stats import norm, skew #for some statistics
import warnings
def ignore_warn(*args, **kwargs):
    pass
warnings.warn = ignore_warn #ignore annoying warning (from sklearn and
seaborn)

#Αρχεία που εμφανίζονται
pd.set_option('display.float_format', lambda x: '{:.3f}'.format(x))
#Πραγματοποιείται περιορισμός των αποτελεσμάτων σε 3 δεκαδικά ψηφία
from subprocess import check_output
print(check_output(["ls", "../input/thesis-2"]).decode("utf8"))
#Πραγματοποιούμε έλεγχο των αρχείων που είναι διαθέσιμα στην πλατφόρμα
```

```
data_description.txt
sample_submission.csv
test.csv
train.csv
```

```
# Πραματοποίηση Εισαγωγής Δεδομένων
train_data = pd.read_csv("../input/thesis-2/train.csv") # Διαβάζει τα
αρχεία του train_data
test_data = pd.read_csv("../input/thesis-2/test.csv") # Διαβάζει τα
αρχεία του test_data
sample_submission_data =
pd.read_csv("../input/thesis-2/sample_submission.csv") # Διαβάζει τα
αρχεία του sample_submission_data

#Στήλες αρχείου train_data
train_data.columns
```

```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea',
'Street',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2',
'BldgType',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt',
'YearRemodAdd',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd',
'MasVnrType',
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation',
```

```
'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF',
'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF',
'2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath',
'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu',
'GarageType',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea',
'GarageQual',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea',
'PoolQC',
       'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold',
'SaleType',
       'SaleCondition', 'SalePrice'],
      dtype='object')
```

```
#Μας ενδιαφέρει η  στήλη SalePrice:
train_data['SalePrice'].describe()
```

```
count   1458.000
mean      12.024
std        0.400
min       10.460
25%       11.775
50%       12.002
75%       12.274
max       13.534
Name: SalePrice, dtype: float64
```

```
#Πραγματοποιείται εμφάνιση των 5 πρώτων γραμμών του dataset train_data
train_data.head(5)
```

```
   Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape
\
0   1          60       RL       65.000     8450   Pave   NaN      Reg

1   2          20       RL       80.000     9600   Pave   NaN      Reg

2   3          60       RL       68.000    11250   Pave   NaN      IR1

3   4          70       RL       60.000     9550   Pave   NaN      IR1

4   5          60       RL       84.000    14260   Pave   NaN      IR1
```

```
  LandContour Utilities  ... PoolArea PoolQC Fence MiscFeature MiscVal
MoSold  \
0         Lvl    AllPub  ...        0    NaN   NaN         NaN       0
2
1         Lvl    AllPub  ...        0    NaN   NaN         NaN       0
5
2         Lvl    AllPub  ...        0    NaN   NaN         NaN       0
9
3         Lvl    AllPub  ...        0    NaN   NaN         NaN       0
2
4         Lvl    AllPub  ...        0    NaN   NaN         NaN       0
12

  YrSold  SaleType  SaleCondition  SalePrice
0   2008        WD         Normal     208500
1   2007        WD         Normal     181500
2   2008        WD         Normal     223500
3   2006        WD        Abnorml     140000
4   2008        WD         Normal     250000

[5 rows x 81 columns]
```

```
#Πραγματοποιείται εμφάνιση των 5 πρώτων γραμμών του dataset test_data
test_data.head(5)
```

```
     Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley
LotShape  \
0  1461          20       RH       80.000    11622   Pave   NaN
Reg
1  1462          20       RL       81.000    14267   Pave   NaN
IR1
2  1463          60       RL       74.000    13830   Pave   NaN
IR1
3  1464          60       RL       78.000     9978   Pave   NaN
IR1
4  1465         120       RL       43.000     5005   Pave   NaN
IR1

  LandContour Utilities  ... ScreenPorch PoolArea PoolQC  Fence
MiscFeature  \
0         Lvl    AllPub  ...         120        0    NaN  MnPrv
NaN
1         Lvl    AllPub  ...           0        0    NaN    NaN
Gar2
2         Lvl    AllPub  ...           0        0    NaN  MnPrv
NaN
3         Lvl    AllPub  ...           0        0    NaN    NaN
NaN
4         HLS    AllPub  ...         144        0    NaN    NaN
NaN
```

```
  MiscVal MoSold  YrSold  SaleType  SaleCondition
0       0      6    2010        WD         Normal
1   12500      6    2010        WD         Normal
2       0      3    2010        WD         Normal
3       0      6    2010        WD         Normal
4       0      1    2010        WD         Normal

[5 rows x 80 columns]
```

```
#Έλεγχος του αριθμού των δειγμάτων και των χαρακτηριστικών
print("Το μέγεθος του train_data πριν από την απόρριψη του
χαρακτηριστικού Id είναι : {}".format(train_data.shape))
print("Το μέγεθος του test_data πριν από την απόρριψη του
χαρακτηριστικού Id είναι : {}".format(test_data.shape))

#Αποθήκευση της στήλης 'Id'
train_data_ID = train_data["Id"]
test_data_ID = test_data["Id"]

#Τώρα διαγράφουμε την στήλη Id διότι μπορεί να θεωρηθεί και όντως
είναι περιττή για την διαδικασία της πρόβλεψης αφού δεν χρησιμεύει
στην ανάλυση του δείγματος##
train_data.drop("Id", axis = 1, inplace = True)
test_data.drop("Id", axis = 1, inplace = True)

#Ξανακάνουμε έλεγχο σχετικά με το μέγεθος των δεδομένων μετά την
αφαίρεση της μεταβλητής 'Id'
print(" \nΤο μέγεθος του train_data μετά την απόρριψη του
χαρακτηριστικού Id είναι {} ".format(train_data.shape))
print("Το μέγεθος του test_data μετά την απόρριψη του χαρακτηριστικού
Id είναι {} ".format(test_data.shape))
```

```
Το μέγεθος του train_data πριν από την απόρριψη του χαρακτηριστικού Id
είναι : (1460, 81)
Το μέγεθος του test_data πριν από την απόρριψη του χαρακτηριστικού Id
είναι : (1459, 80)

Το μέγεθος του train_data μετά την απόρριψη του χαρακτηριστικού Id
είναι (1460, 80)
Το μέγεθος του test_data μετά την απόρριψη του χαρακτηριστικού Id
είναι (1459, 79)
```

```
#Τυποποίηση Δεδομένων - Standardizing data
saleP_scaled = StandardScaler().fit_transform(train_data['SalePrice']
[:,np.newaxis]);
low_range = saleP_scaled[saleP_scaled[:,0].argsort()][:10]
high_range= saleP_scaled[saleP_scaled[:,0].argsort()][-10:]
print('outer range (low) of the distribution:')
print(low_range)
```

```
print('\nouter range (high) of the distribution:')
print(high_range)
```

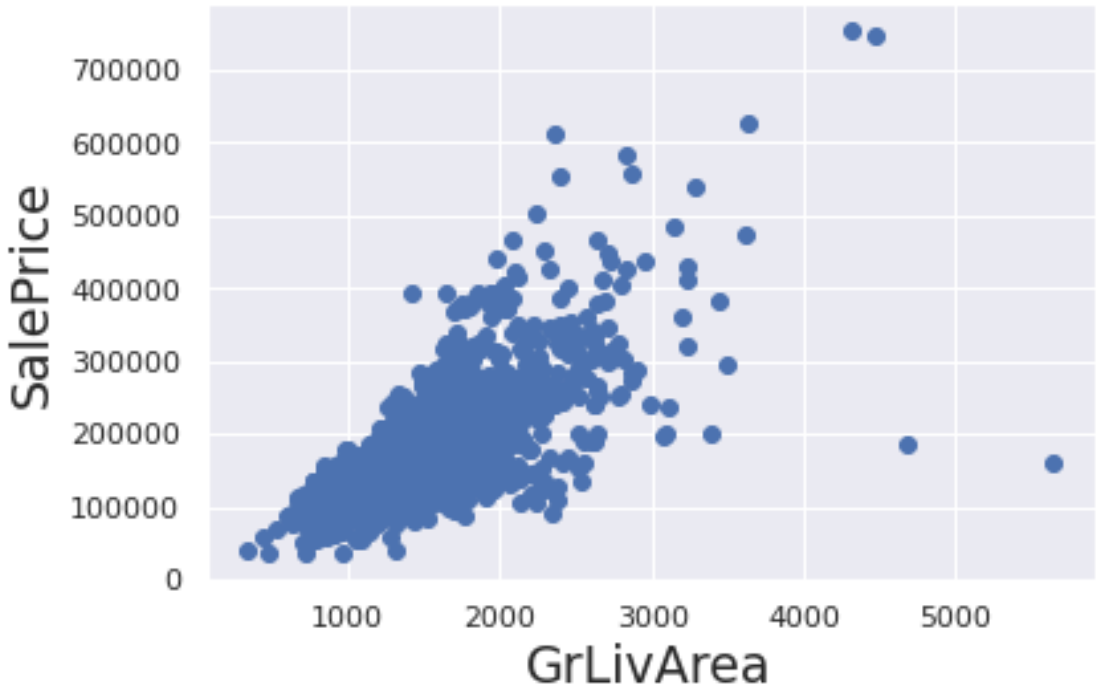
```
outer range (low) of the distribution:
[[-1.83870376]
 [-1.83352844]
 [-1.80092766]
 [-1.78329881]
 [-1.77448439]
 [-1.62337999]
 [-1.61708398]
 [-1.58560389]
 [-1.58560389]
 [-1.5731    ]]

outer range (high) of the distribution:
[[3.82897043]
 [4.04098249]
 [4.49634819]
 [4.71041276]
 [4.73032076]
 [5.06214602]
 [5.42383959]
 [5.59185509]
 [7.10289909]
 [7.22881942]]
```

Προχωράμε σε νέο στάδιο αυτό του Data Processing

Ανακαλύπτωντας Outliers

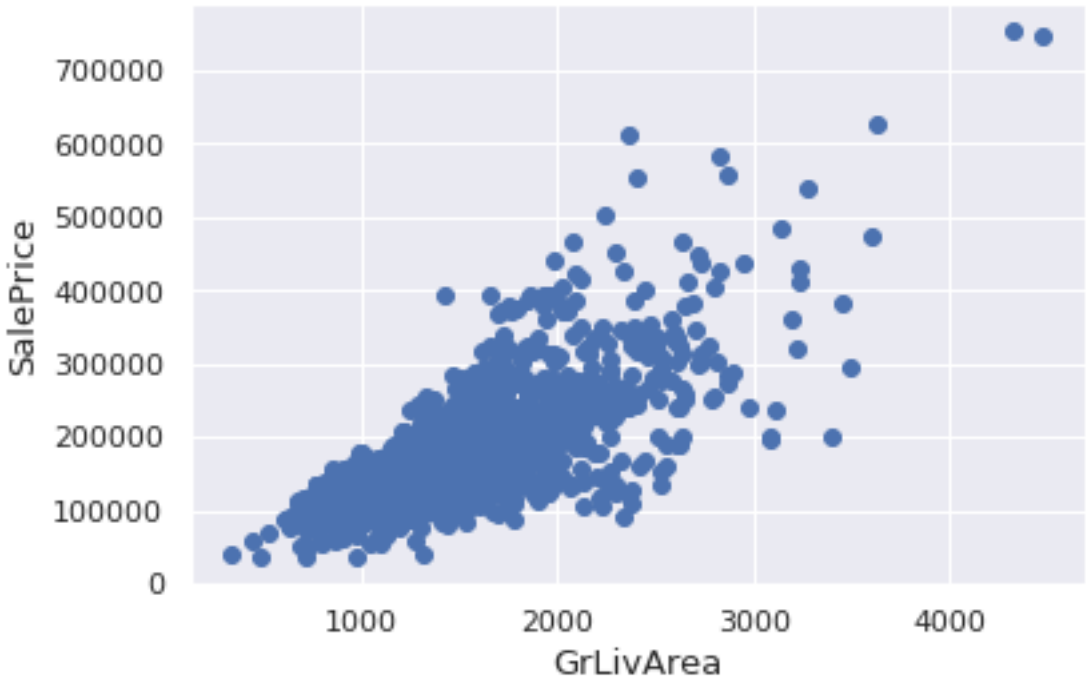
Η ανίχνευση ανωμαλιών (που ονομάζεται επίσης ανίχνευση ακραίων τιμών) αποτελεί έργο ανίχνευσης περιπτώσεων που αποκλίνουν έντονα από τον κανόνα. Αυτές οι περιπτώσεις λοιπόν λέμε ότι ονομάζονται ανωμαλίες ή ακραίες περιπτώσεις.

```
## Ανακαλύπτωντας Outliers
#Η ανίχνευση ανωμαλιών(που ονομάζεται επίσης ανίχνευση ακραίων τιμών)
αποτελεί έργο  ανίχνευσης περιπτώσεων που αποκλίνουν έντονα από τον
κανόνα. Αυτές οι περιπτώσεις λοιπόν
#λέμε ότι ονομάζονται ανωμαλίες ή ακραίες περιπτώσεις.
fig, ax = plt.subplots()
ax.scatter(x = train_data['GrLivArea'], y = train_data['SalePrice'])
plt.ylabel('SalePrice', fontsize=19)
plt.xlabel('GrLivArea', fontsize=19)
plt.show()
```



Σχολιασμός Διαγράμματος Παρατηρούμε κάτω δεξιά στο διάγραμμα δύο σημεία με εξαιρετικά μεγάλη τιμή για την μεταβλητή GrLiveArea οι οποίες όμως έχουν επίσης και χαμηλή τιμή SalePrice Για αυτό τον λόγο τις θεωρούμε ως outliers και ως εκ τούτου τις διαγράφουμε

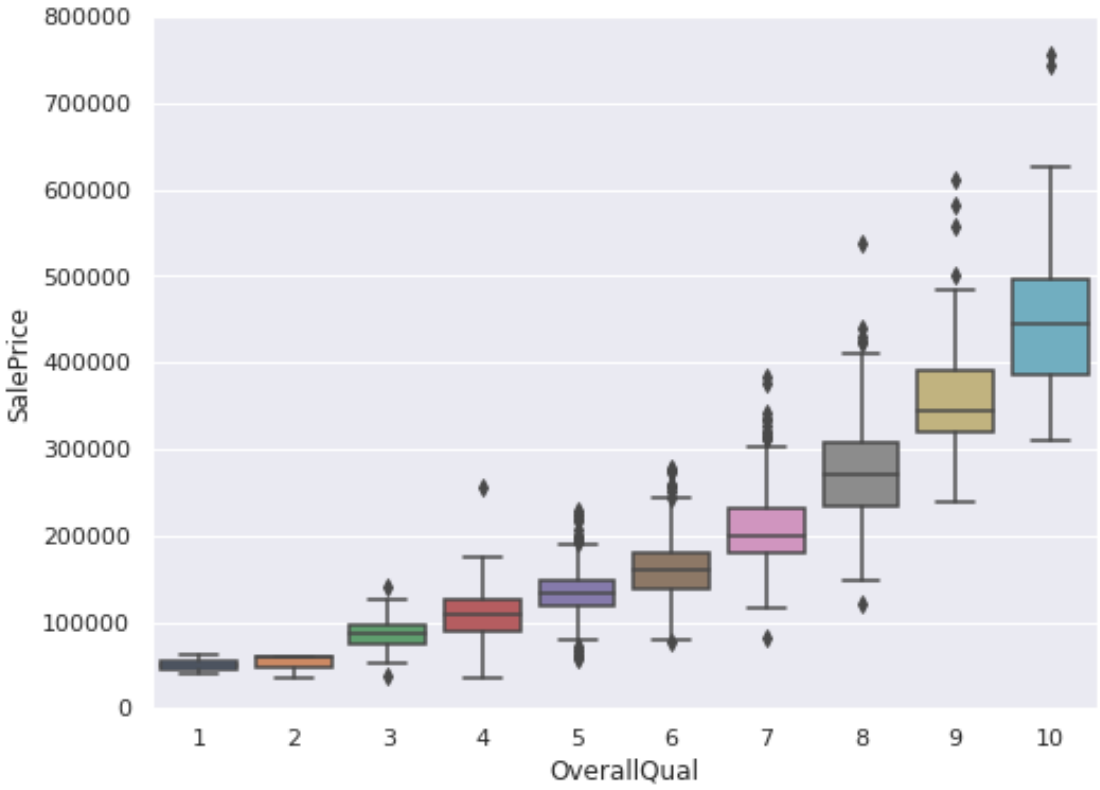
```
#Διαγράφουμε outliers
train_data = train_data.drop(train_data[(train_data['GrLivArea']>4000)
& (train_data['SalePrice']<300000)].index)

#Ξαναελέγχουμε το διάγραμμα
fig, ax = plt.subplots()
ax.scatter(train_data['GrLivArea'], train_data['SalePrice'])
plt.ylabel('SalePrice', fontsize=13)
plt.xlabel('GrLivArea', fontsize=13)
plt.show()
```



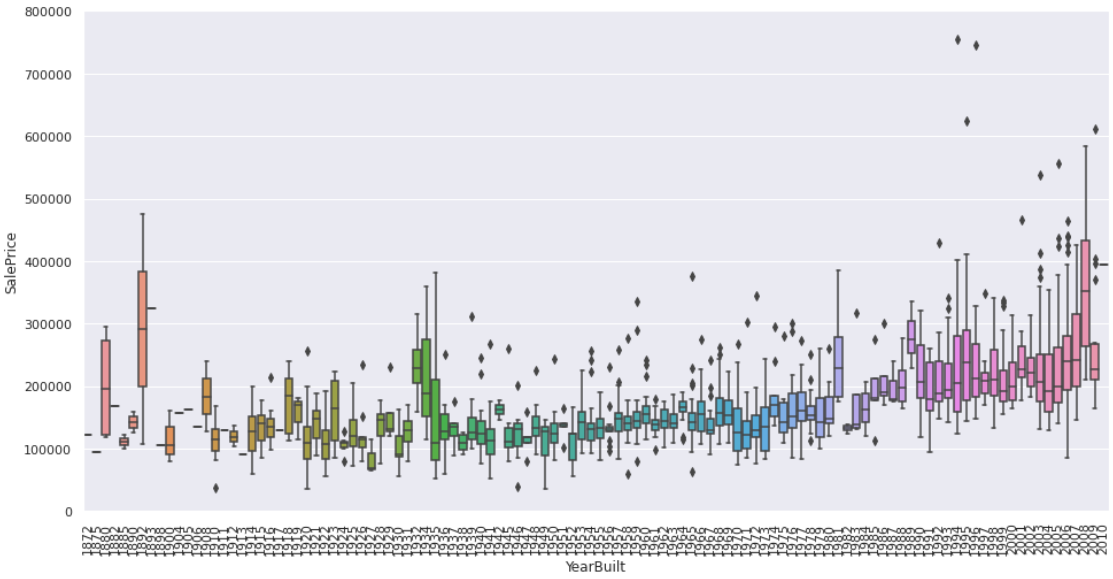
Η αφαίρεση των Outliers πρέπει να είναι πάντα ασφαλής. Αποφασίσαμε να διαγράψουμε αυτά τα δύο σημεία καθώς είναι πολύ μεγάλα.Πιθανόν όμως να υπάρχουν και άλλες ακραίες τιμές στο train_data όπως τα δύο σημεία πάνω δεξιά που φαίνεται να έχουν υπερβολικα υψηλή τιμή για την μεταβλητή GrLiveArea καθώς και υπερβολικά υψηλή τιμή και για την μεταβλητή SalePrice. Ωστόσο, η αφαίρεση όλων αυτών μπορεί να επηρεάσει άσχημα τα μοντέλα μας. Γι' αυτό , αντί να τις αφαιρέσουμε όλες, θα προσπαθήσουμε απλώς να κάνουμε κάποια από τα μοντέλα μας ανθεκτικά σε αυτές.

Η μεταβλητή που μας ενδιαφέρει να προβλέψουμε είναι η SalePrice, για αυτό ξεκινάμε με την ανάλυση αυτής αρχικά.

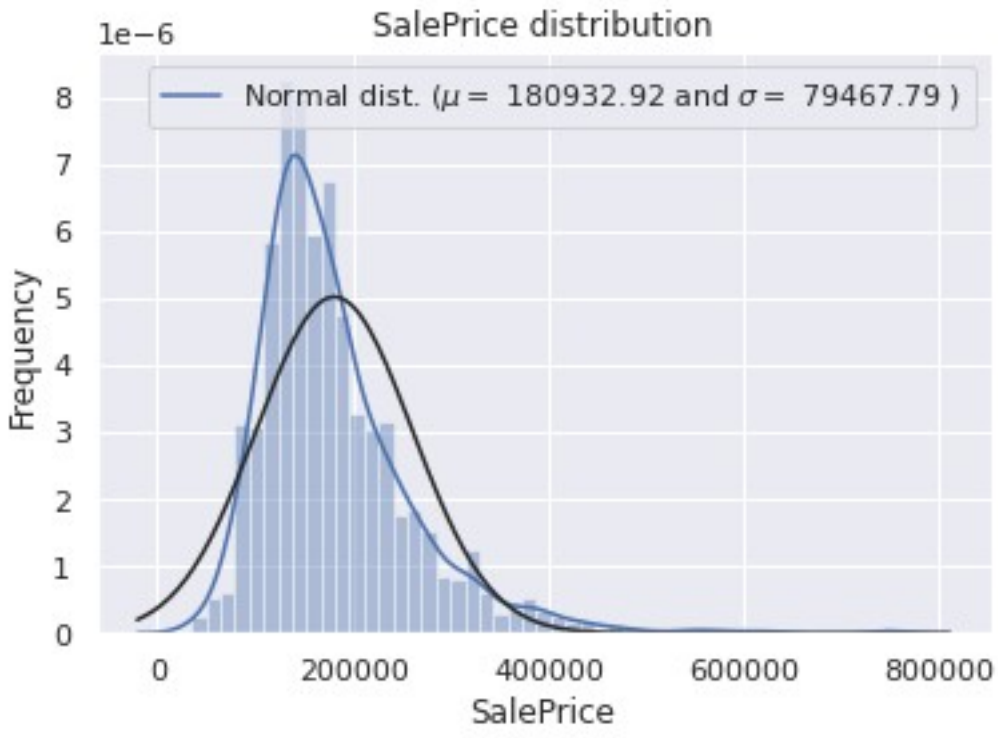
```
# Η Σχέση με τις κατηγορικές μεταβλητές είναι η εξής:
#Παρατηρούμε το εξής: Η 'SalePrice' απολαμβάνει την 'OverallQual'.
#box plot overallqual/saleprice
var = 'OverallQual'
data = pd.concat([train_data['SalePrice'], train_data[var]], axis=1)
f, ax = plt.subplots(figsize=(8, 6))
fig = sns.boxplot(x=var, y="SalePrice", data=data)
fig.axis(ymin=0, ymax=800000);
```




```
# Σχολιασμός Διαγράμματος
#Η μεταβλητή 'SalePrice' έχει θετική συσχέτιση με την μεταβλητή
'OverallQual'.

#Αν και δεν πρόκειται για ισχυρή τάση, θα μπορούσαμε να πούμε ότι η
'SalePrice' είναι πιο επιρρεπής στο να ξοδεύει περισσότερα χρήματα σε
νέα πράγματα παρά σε παλιά.
var = 'YearBuilt'
data = pd.concat([train_data['SalePrice'], train_data[var]], axis=1)
f, ax = plt.subplots(figsize=(16, 8))
fig = sns.boxplot(x=var, y="SalePrice", data=data)
fig.axis(ymin=0, ymax=800000);
plt.xticks(rotation=90);
```




```
# Πριν ξεκινήσουμε να συνθέσουμε τα 2 dataset (train_data and
test_data) θα επιδιώξουμε να βρούμε τις παραμέτρους της κανονικής
κατανομής, οι οποίες είναι η μέση τιμή (mu) και η τυπική απόκλιση
(sigma).


#Διάγραμμα πιθανοτήτων κανονικής κατανομής: η κατανομή των δεδομένων
θα πρέπει να είναι κατά το δυνατόν σύμφωνη με την κανονική κατανομή
sns.distplot(train_data['SalePrice'] , fit=norm);

# Εδώ θα πραγματοποιήσουμε λήψη των προσαρμοσμένων παραμέτρων που
# χρησιμοποιούνται από τη συνάρτηση

(mu, sigma) = norm.fit(train_data['SalePrice'])
print( '\n Μέση τιμή = {:.2f} and Τυπική απόκλιση = {:.2f}\
n'.format(mu, sigma))

# Σχεδιάζουμε την κατανομή
plt.legend(['Normal dist. ($\mu=$ {:.2f} and $\sigma=$
{:.2f} )'.format(mu, sigma)],
            loc='best')
plt.ylabel('Frequency')
plt.title('SalePrice distribution')


 Μέση τιμή = 180932.92 and Τυπική απόκλιση = 79467.79


Text(0.5, 1.0, 'SalePrice distribution')
```

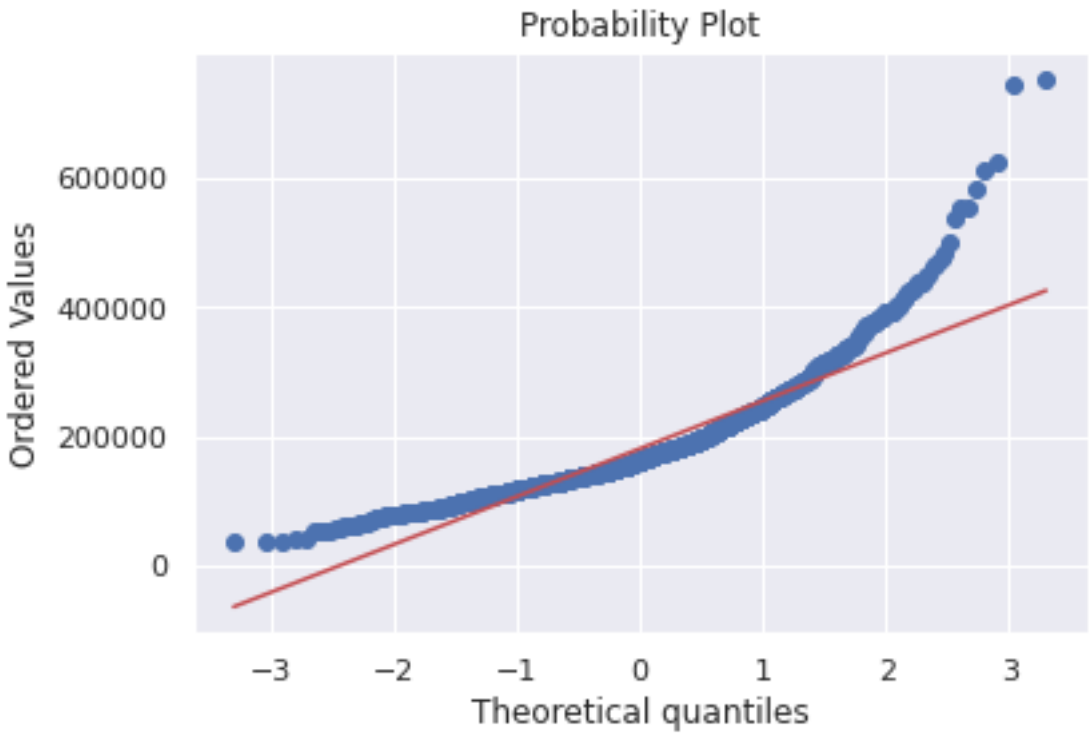


Συμπέρασμα Ιστογράμματος 1.Απόκλιση από την κανονική κατανομής 2.Ύπαρξη αξιοσημείωτης θετικής λοξότητας 3.Εμφάνιση αιχμών

```
#skewness and kurtosis
print("Skewness: %f" % train_data['SalePrice'].skew())
print("Kurtosis: %f" % train_data['SalePrice'].kurt())
```

```
Skewness: 1.881296
Kurtosis: 6.523067
```

```
# Σχεδιάζουμε το Quantile(QQ-plot) _
fig = plt.figure()
res = stats.probplot(train_data['SalePrice'], plot=plt)
plt.show()
```



Τα διαγράμματα ποσοστηρίων χρησιμοποιούνται για την εύρεση του τύπου της κατανομής μιας τυχαίας μεταβλητής, (είτε πρόκειται για κατανομή Gauss,είτε για ομοιόμορφη κατανομή, είτε για εκθετική κατανομή, είτε ακόμη και για κατανομή Pareto)

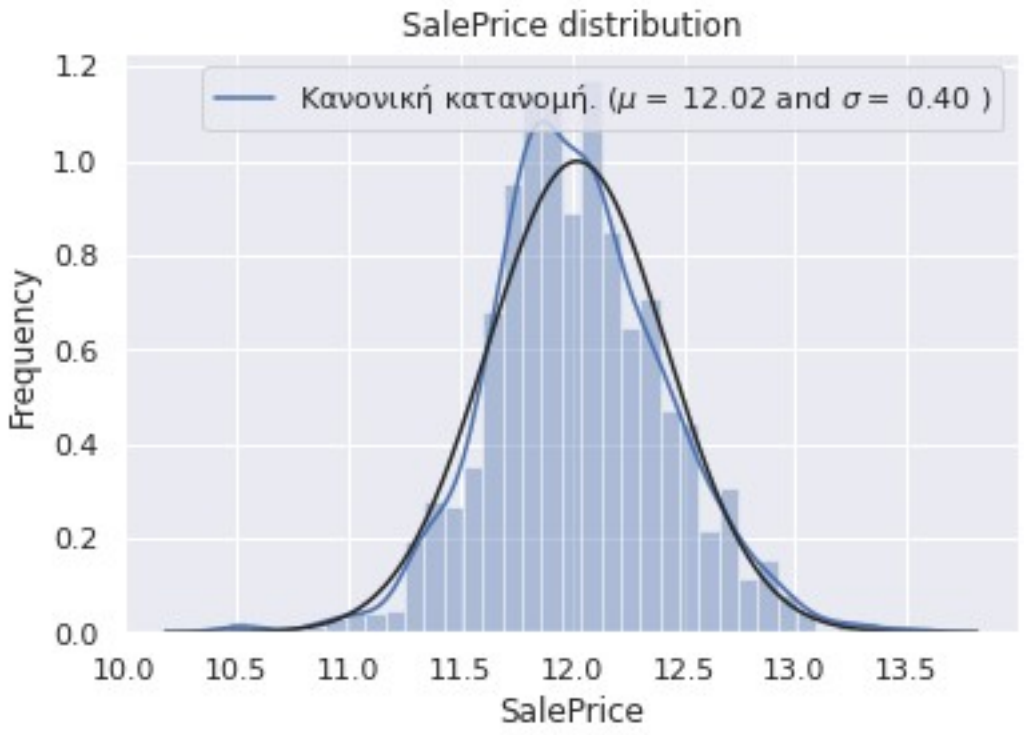
Η μέθοδος για τον έλεγχο της κατανομής πιθανοτήτων των δεδομένων του δείγματος (η προεπιλογή είναι ο έλεγχος της κανονικής κατανομής). Το κόκκινο υποδεικνύει την κανονική κατανομή, το μπλε είναι τα δεδομένα του δείγματος, και όσο πιο κοντά είναι το μπλε, τόσο πιο συνεπής είναι το κόκκινο με την αναμενόμενη κατανομή. Η τιμή πώλησης δεν είναι κανονική κατανομή, αλλά μια περίπτωση κανονικής λοξότητας (δεξιά λοξότητα), η οποία μπορεί να μετασχηματιστεί με λογάριθμο. Η μεταβλητή-στόχος είναι λοξή. Για τον λόγο αυτό θα προσπαθήσουμε να την μετατρέψουμε σε περισσότερο κανονικά κατανεμημένη.

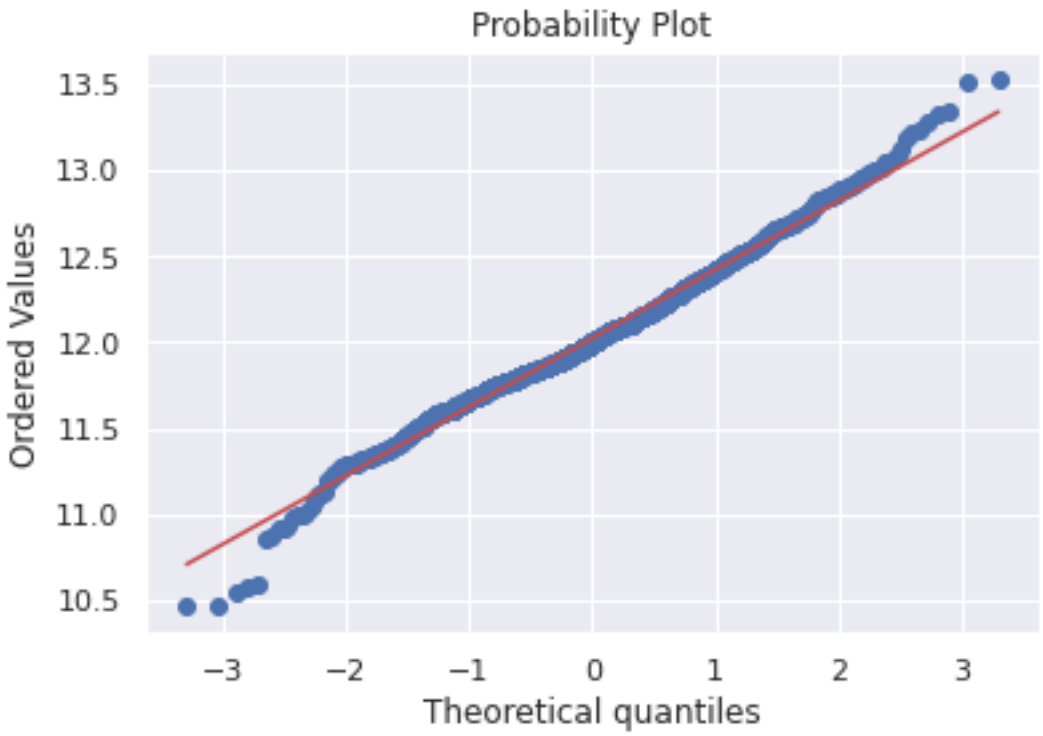
Μετασχηματισμός λογαρίθμου της μεταβλητής-στόχου

```
#Χρησιμοποιούμε τη λειτουργία numpy log1p η οποία εφαρμόζει log(1+x)
σε όλα τα στοιχεία της στήλης
train_data["SalePrice"] = np.log1p(train_data["SalePrice"])

#Έλεγχος της νέας κατανομής
sns.distplot(train_data['SalePrice'] , fit=norm);

# Εδώ θα πραγματοποιήσουμε λήψη των προσαρμοσμένων παραμέτρων που
```

```
χρησιμοποιούνται
#από τη συνάρτηση
(mu, sigma) = norm.fit(train_data['SalePrice'])
print( '\n Μέση τιμή = {:.2f} and Τυπική απόκλιση = {:.2f}\
n'.format(mu, sigma))

#Τώρα σχεδιάζουμε την κατανομή
plt.legend(['Κανονική κατανομή. ($\mu=$ {:.2f} and $\sigma=$
{:.2f} )'.format(mu, sigma)],
            loc='best')
plt.ylabel('Frequency')
plt.title('SalePrice distribution')

# Σχεδιάζουμε το Quantile(QQ-plot) _
fig = plt.figure()
res = stats.probplot(train_data['SalePrice'], plot=plt)
plt.show()
```

```
 Μέση τιμή = 12.02 and Τυπική απόκλιση = 0.40
```





Παρατηρούμε ότι η λοξότητα φαίνεται τώρα να έχει διορθωθεί και τα δεδομένα εμφανίζονται πιο κανονικά κατανεμημένα.

Feature engineering

Ξεκινάμε με την σύνδεση των train_data και των test_data στο ίδιο dataset.

```
ENDTRAIN = train_data.shape[0]
ENDTEST = test_data.shape[0]
y_train = train_data.SalePrice.values
all_the_Data = pd.concat((train_data,
test_data)).reset_index(drop=True)
all_the_Data.drop(['SalePrice'], axis=1, inplace=True)
print(" Το μέγεθος του all_the_Data είναι :
{}".format(all_the_Data.shape))
print(" Το μέγεθος του train_data είναι: {}".format(train_data.shape))
print(" Το μέγεθος του test_data είναι : {}".format(test_data.shape))
```

```
 Το μέγεθος του all_the_Data είναι : (2917, 79)
 Το μέγεθος του train_data είναι: (1458, 80)
 Το μέγεθος του test_data είναι : (1459, 79)
```

```
all_the_Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2917 entries, 0 to 2916
```

```
Data columns (total 79 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   MSSubClass      2917 non-null   int64
 1   MSZoning        2913 non-null   object
 2   LotFrontage     2431 non-null   float64
 3   LotArea         2917 non-null   int64
 4   Street          2917 non-null   object
 5   Alley           198 non-null    object
 6   LotShape        2917 non-null   object
 7   LandContour     2917 non-null   object
 8   Utilities       2915 non-null   object
 9   LotConfig       2917 non-null   object
 10  LandSlope       2917 non-null   object
 11  Neighborhood    2917 non-null   object
 12  Condition1      2917 non-null   object
 13  Condition2      2917 non-null   object
 14  BldgType        2917 non-null   object
 15  HouseStyle      2917 non-null   object
 16  OverallQual     2917 non-null   int64
 17  OverallCond     2917 non-null   int64
 18  YearBuilt       2917 non-null   int64
 19  YearRemodAdd    2917 non-null   int64
 20  RoofStyle       2917 non-null   object
 21  RoofMatl        2917 non-null   object
 22  Exterior1st     2916 non-null   object
 23  Exterior2nd     2916 non-null   object
 24  MasVnrType      2893 non-null   object
 25  MasVnrArea      2894 non-null   float64
 26  ExterQual       2917 non-null   object
 27  ExterCond       2917 non-null   object
 28  Foundation      2917 non-null   object
 29  BsmtQual        2836 non-null   object
 30  BsmtCond        2835 non-null   object
 31  BsmtExposure    2835 non-null   object
 32  BsmtFinType1    2838 non-null   object
 33  BsmtFinSF1      2916 non-null   float64
 34  BsmtFinType2    2837 non-null   object
 35  BsmtFinSF2      2916 non-null   float64
 36  BsmtUnfSF       2916 non-null   float64
 37  TotalBsmtSF     2916 non-null   float64
 38  Heating         2917 non-null   object
 39  HeatingQC       2917 non-null   object
 40  CentralAir      2917 non-null   object
 41  Electrical      2916 non-null   object
 42  1stFlrSF        2917 non-null   int64
 43  2ndFlrSF        2917 non-null   int64
 44  LowQualFinSF    2917 non-null   int64
 45  GrLivArea       2917 non-null   int64
 46  BsmtFullBath    2915 non-null   float64
```

```
 47  BsmtHalfBath   2915 non-null   float64
 48  FullBath       2917 non-null   int64
 49  HalfBath       2917 non-null   int64
 50  BedroomAbvGr   2917 non-null   int64
 51  KitchenAbvGr   2917 non-null   int64
 52  KitchenQual    2916 non-null   object
 53  TotRmsAbvGrd   2917 non-null   int64
 54  Functional     2915 non-null   object
 55  Fireplaces     2917 non-null   int64
 56  FireplaceQu    1497 non-null   object
 57  GarageType     2760 non-null   object
 58  GarageYrBlt    2758 non-null   float64
 59  GarageFinish   2758 non-null   object
 60  GarageCars     2916 non-null   float64
 61  GarageArea     2916 non-null   float64
 62  GarageQual     2758 non-null   object
 63  GarageCond     2758 non-null   object
 64  PavedDrive     2917 non-null   object
 65  WoodDeckSF     2917 non-null   int64
 66  OpenPorchSF    2917 non-null   int64
 67  EnclosedPorch  2917 non-null   int64
 68  3SsnPorch      2917 non-null   int64
 69  ScreenPorch    2917 non-null   int64
 70  PoolArea       2917 non-null   int64
 71  PoolQC         9 non-null      object
 72  Fence          571 non-null    object
 73  MiscFeature    105 non-null    object
 74  MiscVal        2917 non-null   int64
 75  MoSold         2917 non-null   int64
 76  YrSold         2917 non-null   int64
 77  SaleType       2916 non-null   object
 78  SaleCondition  2917 non-null   object
dtypes: float64(11), int64(25), object(43)
memory usage: 1.8+ MB
```

Βρίσκουμε το ποσοστό των ελλειπόντων/χαμένων δεδομένων

```
all_the_Data_na = (all_the_Data.isnull().sum() / len(all_the_Data)) *
100 #Εδώ βρίσκω το ποσοστό
all_the_Data_na = all_the_Data_na.drop(all_the_Data_na[all_the_Data_na
== 0].index).sort_values(ascending=False)[:30] #Εδώ ταξινομώ την
κατάταξη των δεδομένων
missing_data = pd.DataFrame({'Ποσοστό ελλειπόντων' :all_the_Data_na})
missing_data.head(25)
```

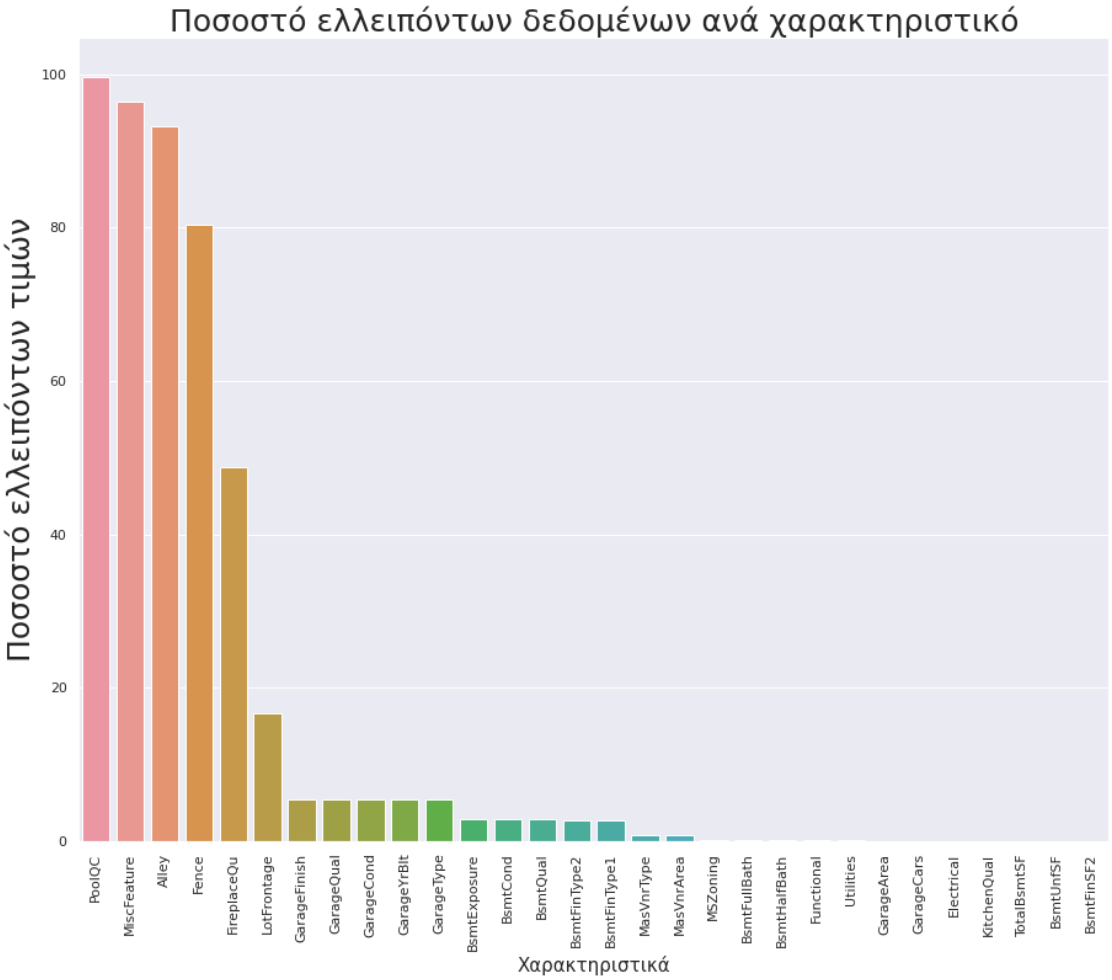
```
              Ποσοστό ελλειπόντων
PoolQC                     99.691
MiscFeature                96.400
Alley                      93.212
Fence                      80.425
```

```
FireplaceQu                48.680
LotFrontage                16.661
GarageFinish                5.451
GarageQual                  5.451
GarageCond                  5.451
GarageYrBlt                 5.451
GarageType                  5.382
BsmtExposure                2.811
BsmtCond                    2.811
BsmtQual                    2.777
BsmtFinType2                2.743
BsmtFinType1                2.708
MasVnrType                  0.823
MasVnrArea                  0.788
MSZoning                    0.137
BsmtFullBath                0.069
BsmtHalfBath                0.069
Functional                  0.069
Utilities                   0.069
GarageArea                  0.034
GarageCars                  0.034
```

Θέλω να απεικονίσω τώρα τα ελλειπή δεδομένα

```
f, ax = plt.subplots(figsize=(15, 12))
plt.xticks(rotation='90')
sns.barplot(x=all_the_Data_na.index, y=all_the_Data_na)
plt.xlabel('Χαρακτηριστικά', fontsize=15)
plt.ylabel('Πσοστό ελλειπόντων τιμών', fontsize=25)
plt.title('Πσοστό ελλειπόντων δεδομένων ανά χαρακτηριστικό',
fontsize=25)
```

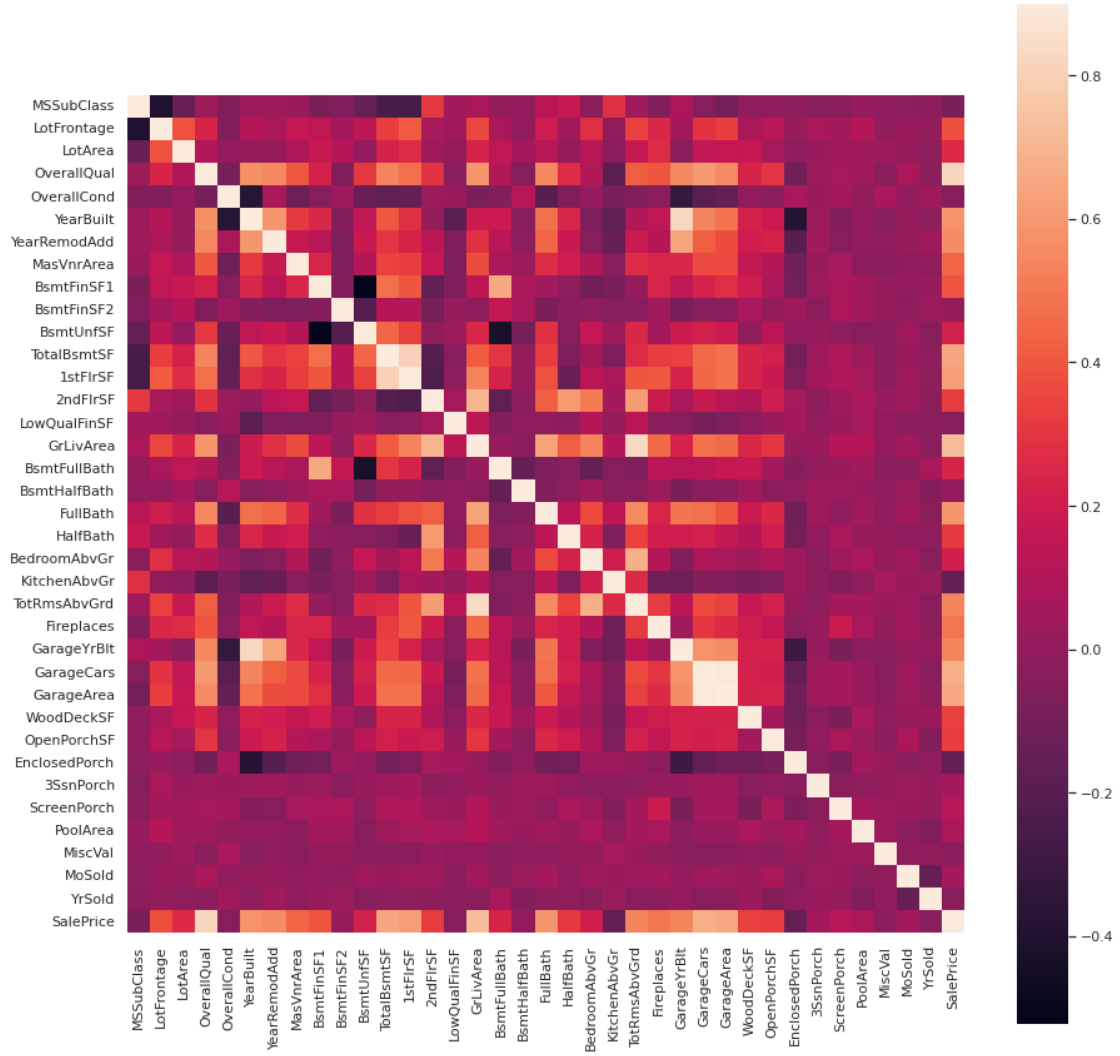
```
Text(0.5, 1.0, 'Πσοστό ελλειπόντων δεδομένων ανά χαρακτηριστικό')
```



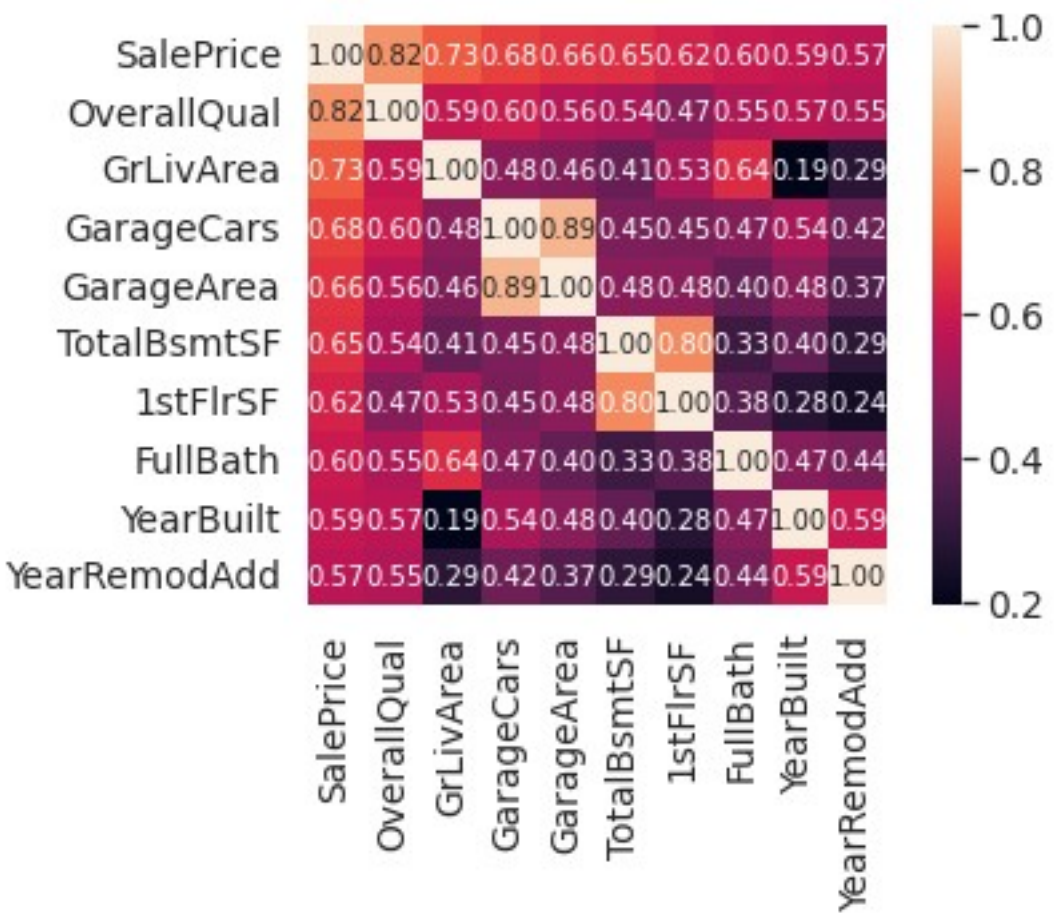
**** Βρίσκω τον χάρτη συσχέτισης έτσι ώστε να δω πώς συσχετίζονται τα χαρακτηριστικά με την τιμή πώλησης

```
corrmat = train_data.corr()
plt.subplots(figsize=(15,15))
sns.heatmap(corrmat, vmax=0.9, square=True)
```

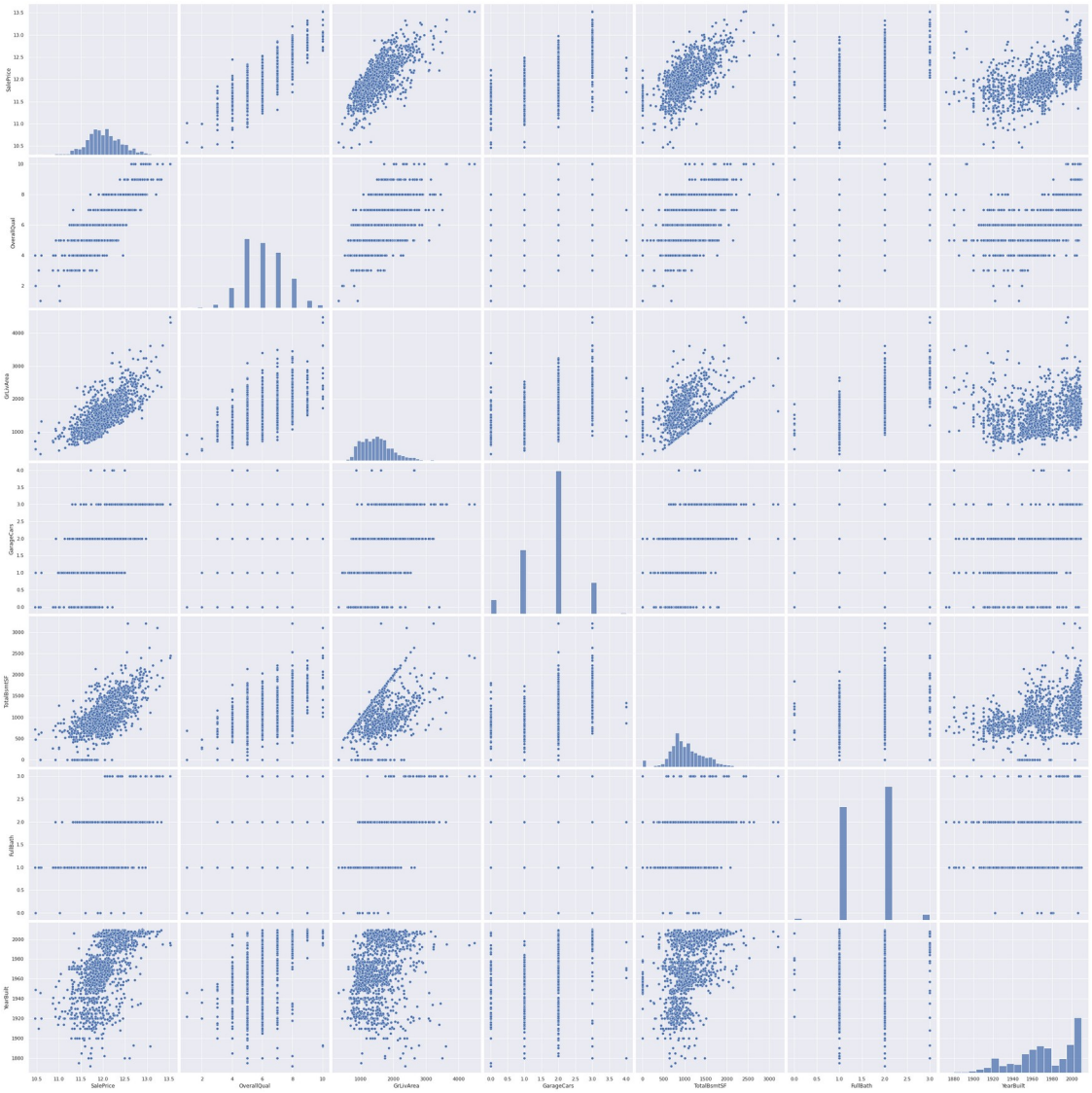
```
<AxesSubplot:>
```




```
#'SalePrice' correlation matrix (Zoomed Heatmap Style)

k = 10 #number of variables for heatmap
cols = corrmat.nlargest(k, 'SalePrice')['SalePrice'].index
cm = np.corrcoef(train_data[cols].values.T)
sns.set(font_scale=1.25)
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f',
annot_kws={'size': 10}, yticklabels=cols.values,
xticklabels=cols.values)
plt.show()
```




```
#Εδώ παρατηρούμε ένα συνολικό scatterplot
sns.set()
cols = ['SalePrice', 'OverallQual', 'GrLivArea', 'GarageCars',
'TotalBsmtSF', 'FullBath', 'YearBuilt']
sns.pairplot(train_data[cols], size = 5)
plt.show();
```



Τώρα μπορούμε να ξεκινήσουμε τον υπολογισμό των ελλιπών τιμών και θα αρχίσουμε ως εξής: Τις υπολογίζουμε προχωρώντας διαδοχικά σε χαρακτηριστικά με ελλειπούσες τιμές.

**Αυτά τα χαρακτηριστικά/μεταβλητές που θα αναλύσουμε παρακάτων είναι οι εξής:

1. PoolQC
2. MiscFeature
3. Alley
4. Fence
5. FireplaceQu
6. LotFrontage

Η περιγραφή των δεδομένων λέει ότι NA σημαίνει "No Pool" - δηλαδή πως δεν υπάρχει πισίνα. Αυτό άλλωστε είναι και λογικό, δεδομένης

της τεράστιας αναλογίας των τιμών που λείπουν (99%) και της πλειοψηφίας των σπιτιών που δεν έχουν καθόλου πισίνα γενικά.

```
#PoolQC
all_the_Data["PoolQC"] = all_the_Data["PoolQC"].fillna("None")
```

MiscFeature : Η περιγραφή των δεδομένων λέει ότι NA σημαίνει "no misc feature".

```
#MissFeature
all_the_Data["MiscFeature"] =
all_the_Data["MiscFeature"].fillna("None")
```

Alley : Η περιγραφή δεδομένων λέει ότι NA σημαίνει "no alley".

```
#Alley
all_the_Data["Alley"] = all_the_Data["Alley"].fillna("None")
```

Fence : η περιγραφή των δεδομένων λέει ότι NA σημαίνει "noFence".

```
#Fence
all_the_Data["Fence"] = all_the_Data["Fence"].fillna("None")
```

#FireplaceQu : η περιγραφή δεδομένων λέει ότι NA σημαίνει "no fireplace".

```
#FireplaceQu
all_the_Data["FireplaceQu"] =
all_the_Data["FireplaceQu"].fillna("None")
```

**LotFrontage : Δεδομένου ότι η περιοχή κάθε δρόμου που συνδέεται με την ιδιοκτησία του σπιτιού

#πιθανότατα έχει παρόμοια επιφάνεια με άλλα σπίτια στη γειτονιά του,

#μπορούμε να συμπληρώσουμε τις τιμές που λείπουν με τη διάμεση τιμή LotFrontage της γειτονιάς.**

```
all_the_Data["LotFrontage"].mode()
```

```
0   60.000
dtype: float64
```

```
#LotFrontage

all_the_Data["LotFrontage"] = all_the_Data.groupby("Neighborhood")
["LotFrontage"].transform(
    lambda x: x.fillna(x.median()))

#GarageType, GarageFinish, GarageQual και GarageCond : Αντικατάσταση
ελλειπόντων δεδομένων με None
```

```
for col in ('GarageType', 'GarageFinish', 'GarageQual', 'GarageCond'):
    all_the_Data[col] = all_the_Data[col].fillna('None')

#GarageYrBlt, GarageArea και GarageCars : Αντικατάσταση των δεδομένων
που λείπουν με 0 (Δεδομένου ότι δεν υπάρχει γκαράζ = δεν υπάρχουν
αυτοκίνητα στο εν λόγω γκαράζ).
for col in ('GarageYrBlt', 'GarageArea', 'GarageCars'):
    all_the_Data[col] = all_the_Data[col].fillna(0)

#BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, TotalBsmtSF, BsmtFullBath και
BsmtHalfBath : οι τιμές που λείπουν είναι πιθανότατα μηδενικές επειδή
δεν υπάρχει υπόγειο.
for col in ('BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF','TotalBsmtSF',
'BsmtFullBath', 'BsmtHalfBath'):
    all_the_Data[col] = all_the_Data[col].fillna(0)

##BsmtQual, BsmtCond, BsmtExposure, BsmtFinType1 και BsmtFinType2 :
Για όλα αυτά τα κατηγορικά χαρακτηριστικά που σχετίζονται με το
υπόγειο,
#το NaN σημαίνει ότι δεν υπάρχει υπόγειο.
for col in ('BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1',
'BsmtFinType2'):
    all_the_Data[col] = all_the_Data[col].fillna('None')

##MasVnrArea και MasVnrType : NA σημαίνει πιθανότατα ότι δεν υπάρχει
τοιχοποιία για αυτά τα σπίτια. Μπορούμε να συμπληρώσουμε 0 για την
περιοχή και Κανένα
#για τον τύπο.
all_the_Data["MasVnrType"] = all_the_Data["MasVnrType"].fillna("None")
all_the_Data["MasVnrArea"] = all_the_Data["MasVnrArea"].fillna(0)

#MSZoning (Η γενική ταξινόμηση ζωνών) : Η τιμή "RL" είναι μακράν η πιο
συνηθισμένη. Έτσι, μπορούμε να συμπληρώσουμε τις τιμές που λείπουν με
'RL'.
all_the_Data['MSZoning'] =
all_the_Data['MSZoning'].fillna(all_the_Data['MSZoning'].mode()[0])

#Utilities : Για αυτό το κατηγορηματικό χαρακτηριστικό όλες οι
εγγραφές είναι "AllPub", εκτός από μία "NoSeWa" και 2 NA .
#Δεδομένου ότι το σπίτι με το "NoSewa" βρίσκεται στο σύνολο
εκπαίδευσης,
#αυτό το χαρακτηριστικό δεν θα βοηθήσει στην προγνωστική
μοντελοποίηση. Μπορούμε λοιπόν να το αφαιρέσουμε με ασφάλεια.
all_the_Data = all_the_Data.drop(['Utilities'], axis=1)

#Functional : η περιγραφή των δεδομένων λέει ότι NA σημαίνει τυπικό
all_the_Data["Functional"] = all_the_Data["Functional"].fillna("Typ")

#Electrical : Έχει μία τιμή NA. Δεδομένου ότι αυτό το χαρακτηριστικό
έχει ως επί το πλείστον 'SBrkr', μπορούμε να το ορίσουμε για την τιμή
που λείπει.
all_the_Data['Electrical'] =
```

```
all_the_Data['Electrical'].fillna(all_the_Data['Electrical'].mode()
[0])

#KitchenQual: (η οποία είναι η πιο συχνή) για την τιμή που λείπει στο
KitchenQual.
all_the_Data['KitchenQual'] =
all_the_Data['KitchenQual'].fillna(all_the_Data['KitchenQual'].mode()
[0])

#Exterior1st και Exterior2nd : Και πάλι τόσο το Exterior 1 όσο και το
Exterior 2 έχουν μόνο
#μία τιμή που λείπει. Θα αντικαταστήσουμε την πιο κοινή συμβολοσειρά
all_the_Data['Exterior1st'] =
all_the_Data['Exterior1st'].fillna(all_the_Data['Exterior1st'].mode()
[0])
all_the_Data['Exterior2nd'] =
all_the_Data['Exterior2nd'].fillna(all_the_Data['Exterior2nd'].mode()
[0])

#SaleType : Συμπληρώνω ξανά με το πιο συχνό που είναι το "WD".
all_the_Data['Exterior1st'] =
all_the_Data['Exterior1st'].fillna(all_the_Data['Exterior1st'].mode()
[0])
all_the_Data['Exterior2nd'] =
all_the_Data['Exterior2nd'].fillna(all_the_Data['Exterior2nd'].mode()
[0])

#MSSubClass : Το Na πιθανότατα σημαίνει No building class. Μπορούμε να
αντικαταστήσουμε τις ελλείπουσες τιμές με None
all_the_Data['MSSubClass'] = all_the_Data['MSSubClass'].fillna("None")

#Έλεγχος των υπόλοιπων τιμών που λείπουν, εάν υπάρχουν
all_the_Data_na = (all_the_Data.isnull().sum() / len(all_the_Data)) *
100
all_the_Data_na = all_the_Data_na.drop(all_the_Data_na[all_the_Data_na
== 0].index).sort_values(ascending=False)
missing_data = pd.DataFrame({'Πσοστό έλλειψης' :all_the_Data_na})
missing_data.head()
```

	Πσοστό έλλειψης
SaleType	0.034

Δεν εξακολουθεί να λείπει καμία τιμή.

Μετασχηματισμός ορισμένων αριθμητικών μεταβλητών που στην πραγματικότητα είναι κατηγορικές

```
#MSSubClass=The building class
all_the_Data['MSSubClass'] = all_the_Data['MSSubClass'].apply(str)


#Changing OverallCond into a categorical variable
```

```
all_the_Data['OverallCond'] = all_the_Data['OverallCond'].astype(str)


#Year and month sold are transformed into categorical features.
all_the_Data['YrSold'] = all_the_Data['YrSold'].astype(str)
all_the_Data['MoSold'] = all_the_Data['MoSold'].astype(str)
```

Label Encoding, Ετικέτα Κωδικοποίηση ορισμένων κατηγορικών μεταβλητών που μπορεί να περιέχουν πληροφορίες στο σύνολο της διάταξής τους

```
from sklearn.preprocessing import LabelEncoder
cols = ('FireplaceQu', 'BsmtQual', 'BsmtCond', 'GarageQual',
'GarageCond',
        'ExterQual', 'ExterCond','HeatingQC', 'PoolQC', 'KitchenQual',
'BsmtFinType1',
        'BsmtFinType2', 'Functional', 'Fence', 'BsmtExposure',
'GarageFinish', 'LandSlope',
        'LotShape', 'PavedDrive', 'Street', 'Alley', 'CentralAir',
'MSSubClass', 'OverallCond',
        'YrSold', 'MoSold')
# process columns, apply LabelEncoder to categorical features
for c in cols:
    lbl = LabelEncoder()
    lbl.fit(list(all_the_Data[c].values))
    all_the_Data[c] = lbl.transform(list(all_the_Data[c].values))

# shape
print('Shape all_the_Data: {}'.format(all_the_Data.shape))
```

```
Shape all_the_Data: (2917, 78)
```

Προσθήκη ενός ακόμη σημαντικού χαρακτηριστικού

Δεδομένου ότι τα χαρακτηριστικά που σχετίζονται με το εμβαδόν είναι πολύ σημαντικά για τον προσδιορισμό των τιμών των κατοικιών, προσθέτουμε ένα ακόμη χαρακτηριστικό που είναι το συνολικό εμβαδόν του υπογείου, του πρώτου και του δεύτερου ορόφου κάθε κατοικίας

```
## Adding total sqfootage feature
all_the_Data['TotalSF'] = all_the_Data['TotalBsmtSF'] +
all_the_Data['1stFlrSF'] + all_the_Data['2ndFlrSF']
```

Λοξά χαρακτηριστικά / Skewed features

```
numeric_feats = all_the_Data.dtypes[all_the_Data.dtypes != "object"].index

# Έλεγχος της λοξότητας όλων των αριθμητικών χαρακτηριστικών
skewed_feats = all_the_Data[numeric_feats].apply(lambda x: skew(x.dropna())).sort_values(ascending=False)
print("\nΛοξότητα σε αριθμητικά χαρακτηριστικά: \n")
skewness = pd.DataFrame({'Λοξότητα' :skewed_feats})
skewness.head(10)
```

```
Λοξότητα σε αριθμητικά χαρακτηριστικά:
```

	Λοξότητα
MiscVal	21.940
PoolArea	17.689
LotArea	13.109
LowQualFinSF	12.085
3SsnPorch	11.372
LandSlope	4.973
KitchenAbvGr	4.301
BsmtFinSF2	4.145
EnclosedPorch	4.002
ScreenPorch	3.945

Μετασχηματισμός Box Cox ιδιαίτερα λοξών χαρακτηριστικών (highly) skewed features

Χρησιμοποιούμε τη συνάρτηση boxcox1p της scipy η οποία υπολογίζει τον μετασχηματισμό Box-Cox του 1+x .

Σημειώνεται ότι ο ορισμός =0 είναι ισοδύναμος με το log1p που χρησιμοποιήθηκε παραπάνω για την μεταβλητή-στόχο.

```
skewness = skewness[abs(skewness) > 0.75]
print("There are {} skewed numerical features to Box Cox transform".format(skewness.shape[0]))

from scipy.special import boxcox1p
skewed_features = skewness.index
lam = 0.15
for feat in skewed_features:
    #all_the_Data[feat] += 1
    all_the_Data[feat] = boxcox1p(all_the_Data[feat], lam)
```

```
all_the_Data[skewed_features] =
np.log1p(all_the_Data[skewed_features])

There are 59 skewed numerical features to Box Cox transform

all_the_Data.head()

   MSSubClass MSZoning  LotFrontage  LotArea  Street  Alley  LotShape
\
0       1.357       RL        1.922    3.006   0.548  0.548     0.933

1       1.117       RL        1.977    3.031   0.548  0.548     0.933

2       1.357       RL        1.934    3.061   0.548  0.548     0.000

3       1.389       RL        1.900    3.030   0.548  0.548     0.000

4       1.357       RL        1.990    3.106   0.548  0.548     0.000


  LandContour LotConfig  LandSlope  ... PoolArea PoolQC Fence
MiscFeature  \
0         Lvl    Inside      0.000  ...    0.000  0.933 1.037
None
1         Lvl       FR2      0.000  ...    0.000  0.933 1.037
None
2         Lvl    Inside      0.000  ...    0.000  0.933 1.037
None
3         Lvl    Corner      0.000  ...    0.000  0.933 1.037
None
4         Lvl       FR2      0.000  ...    0.000  0.933 1.037
None

  MiscVal  MoSold  YrSold  SaleType  SaleCondition TotalSF
0   0.000   1.037   0.786        WD         Normal   2.771
1   0.000   1.236   0.548        WD         Normal   2.768
2   0.000   1.389   0.786        WD         Normal   2.782
3   0.000   1.037   0.000        WD        Abnorml   2.764
4   0.000   0.933   0.786        WD         Normal   2.824

[5 rows x 79 columns]
```

Getting dummy categorical features

```
all_the_Data = pd.get_dummies(all_the_Data)
print(all_the_Data.shape)

(2917, 220)
```

Απόκτηση των νέων συνόλων εκπαίδευσης και δοκιμών.

```
endtrain = all_the_Data[:ENDTRAIN]
endtest = all_the_Data[ENDTRAIN:]
```

Πραγματοποίηση μοντελοποίσησης

Προκειμένω να συνεχιστεί ο έλεγχος, είναι ανάγκη να κάνουμε εισαγωγή συγκεκριμένες βιβλιοθήκες της sklearn.

```
from sklearn.linear_model import ElasticNet, Lasso,  BayesianRidge,
LassoLarsIC
from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor
from sklearn.kernel_ridge import KernelRidge
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import RobustScaler
from sklearn.base import BaseEstimator, TransformerMixin,
RegressorMixin, clone
from sklearn.model_selection import KFold, cross_val_score,
train_test_split
from sklearn.metrics import mean_squared_error
import xgboost as xgb
import lightgbm as lgb
```

Μερικές έννοιες εκ των βιβλιοθηκών που αναλύονται παρακάτω είναι οι εξής: Η βιβλιοθήκη Sklearn.linear_model είναι μια κλάση της ενότητας sklearn που περιέχει διάφορες συναρτήσεις για την εκτέλεση μηχανικής μάθησης με γραμμικά μοντέλα. Ο όρος γραμμικό μοντέλο υπονοεί ότι το μοντέλο καθορίζεται ως γραμμικός συνδυασμός χαρακτηριστικών.

Τι είναι το Elastic Net; Η γραμμική παλινδρόμηση με Elastic Net χρησιμοποιεί/συνδυάζει τις ποινές από τις τεχνικές lasso και ridge για την κανονικοποίηση των μοντέλων παλινδρόμησης.

Τι είναι το Lasso στο Sklearn παλινδρόμηση Lasso χρησιμοποιείται στη μηχανική μάθηση για την αποφυγή της υπερπροσαρμογής.

Παλινδρόμηση Ridge Η παλινδρόμηση Ridge χρησιμοποιεί κανονικοποίηση με τη νόρμα L2, ενώ η Bayesian παλινδρόμηση, είναι ένα μοντέλο παλινδρόμησης που ορίζεται με πιθανολογικούς όρους, με ρητές προτεραιότητες για τις παραμέτρους. Δεν είναι το ίδιο, διότι η παλινδρόμηση κορυφογραμμής είναι ένα είδος μοντέλου παλινδρόμησης και η Μπεϋζιανή προσέγγιση είναι ένας γενικός τρόπος ορισμού και εκτίμησης στατιστικών μοντέλων που μπορεί να εφαρμοστεί σε διαφορετικά μοντέλα.

Προσαρμογή μοντέλου Lasso με Lars χρησιμοποιώντας BIC ή AIC για την επιλογή μοντέλου.

Ο στόχος βελτιστοποίησης για το Lasso είναι: AIC είναι το κριτήριο πληροφοριών Akaike και BIC είναι το κριτήριο πληροφοριών Bayes. Τα κριτήρια αυτά είναι χρήσιμα για την επιλογή της τιμής της παραμέτρου κανονικοποίησης κάνοντας έναν συμβιβασμό μεταξύ της καλής προσαρμογής και της πολυπλοκότητας του μοντέλου. Ένα καλό μοντέλο θα πρέπει να εξηγεί καλά τα δεδομένα και ταυτόχρονα να είναι απλό.

Σε αυτό το σημείο πριν ξεκινήσουμε θα δημιουργήσουμε/θέσουμε μια στρατηγική διασταυρούμενης επικύρωσης(cross validation)

Χρησιμοποιούμε τη συνάρτηση **cross_val_score** του Sklearn. Ωστόσο, αυτή η συνάρτηση δεν έχει χαρακτηριστικό όπως θα το θέλαμε(shuffled), προσθέτουμε λοιπόν μια γραμμή κώδικα, προκειμένου να ανακατέψουμε το σύνολο δεδομένων πριν από τη διασταυρούμενη επικύρωση.

```
#Validation function
n_folds = 5

def rmsle_cv(model):
    kf = KFold(n_folds, shuffle=True,
random_state=42).get_n_splits(endtrain.values)
    rmse= np.sqrt(-cross_val_score(model, endtrain.values, y_train,
scoring="neg_mean_squared_error", cv = kf))
    return(rmse)
```

ΒΑΣΙΚΑ ΜΟΝΤΕΛΑ / BASIC MODELS

Lasso, Elastic Net Regression, Kernel Ridge Regression, Gradient Boosting Regression,XGBoost, LightGBM

Παλινδρόμηση LASSO : Αυτό το μοντέλο μπορεί να είναι πολύ ευαίσθητο στις ακραίες τιμές. Έτσι πρέπει να το κάνουμε πιο ανθεκτικό σε αυτές. Για το σκοπό αυτό χρησιμοποιούμε τη μέθοδο Robustscaler() του sklearn στην pipeline.

```
lasso = make_pipeline(RobustScaler(), Lasso(alpha = 0.0005,
random_state=1))

score = rmsle_cv(lasso)
print("\nLasso score: {:.4f} ({:.4f})\n".format(score.mean(),
score.std()))


Lasso score: 0.1148 (0.0086)
```

Elastic Net Regression και πάλι πιο ανθεκτική στις ακραίες τιμές

```
ENet = make_pipeline(RobustScaler(), ElasticNet(alpha=0.0005,
l1_ratio=.9, random_state=3))
```

```
myscore = rmsle_cv(ENet)
print("ElasticNet score: {:.4f} ({:.4f})\n".format(myscore.mean(),
myscore.std()))
```

```
ElasticNet score: 0.1148 (0.0086)
```

Kernel Ridge Regression

```
KRR = KernelRidge(alpha=0.6, kernel='polynomial', degree=2, coef0=2.5)

score = rmsle_cv(KRR)
print("Kernel Ridge score: {:.4f} ({:.4f})\n".format(score.mean(),
score.std()))
```

```
Kernel Ridge score: 0.1439 (0.0075)
```

Gradient Boosting Regression: Με απώλειες Huber που το καθιστούν ανθεκτικό στις ακραίες τιμές

```
GBoost = GradientBoostingRegressor(n_estimators=3000,
learning_rate=0.05,
                                   max_depth=4, max_features='sqrt',
                                   min_samples_leaf=15,
min_samples_split=10,
                                   loss='huber', random_state =5)

score = rmsle_cv(GBoost)
print("Gradient Boosting score: {:.4f} ({:.4f})\
n".format(score.mean(), score.std()))
```

```
Gradient Boosting score: 0.1167 (0.0083)
```

XGBoost

```
model_xgb = xgb.XGBRegressor(colsample_bytree=0.4603, gamma=0.0468,
                             learning_rate=0.05, max_depth=3,
                             min_child_weight=1.7817,
n_estimators=2200,
                             reg_alpha=0.4640, reg_lambda=0.8571,
                             subsample=0.5213, silent=1,
                             random_state =7, nthread = -1)

score = rmsle_cv(model_xgb)
print("Xgboost score: {:.4f} ({:.4f})\n".format(score.mean(),
score.std()))
```

```
[05:18:33] WARNING: ../src/learner.cc:627:
Parameters: { "silent" } might not be used.

  This could be a false alarm, with some parameters getting used by
language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter
actually being used
  but getting flagged wrongly here. Please open an issue if you find
any such cases.


[05:18:47] WARNING: ../src/learner.cc:627:
Parameters: { "silent" } might not be used.

  This could be a false alarm, with some parameters getting used by
language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter
actually being used
  but getting flagged wrongly here. Please open an issue if you find
any such cases.


[05:18:59] WARNING: ../src/learner.cc:627:
Parameters: { "silent" } might not be used.

  This could be a false alarm, with some parameters getting used by
language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter
actually being used
  but getting flagged wrongly here. Please open an issue if you find
any such cases.


[05:19:12] WARNING: ../src/learner.cc:627:
Parameters: { "silent" } might not be used.

  This could be a false alarm, with some parameters getting used by
language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter
actually being used
  but getting flagged wrongly here. Please open an issue if you find
any such cases.


[05:19:25] WARNING: ../src/learner.cc:627:
Parameters: { "silent" } might not be used.

  This could be a false alarm, with some parameters getting used by
language bindings but
```

```
  then being mistakenly passed down to XGBoost core, or some parameter
actually being used
  but getting flagged wrongly here. Please open an issue if you find
any such cases.


Xgboost score: 0.1172 (0.0050)
```

LightGBM:

```
model_lgb = lgb.LGBMRegressor(objective='regression',num_leaves=5,
                              learning_rate=0.05, n_estimators=720,
                              max_bin = 55, bagging_fraction = 0.8,
                              bagging_freq = 5, feature_fraction =
0.2319,
                              feature_fraction_seed=9, bagging_seed=9,
                              min_data_in_leaf =6,
min_sum_hessian_in_leaf = 11)

score = rmsle_cv(model_lgb)
print("LGBM score: {:.4f} ({:.4f})\n" .format(score.mean(),
score.std()))

[LightGBM] [Warning] feature_fraction is set=0.2319,
colsample_bytree=1.0 will be ignored. Current value:
feature_fraction=0.2319
[LightGBM] [Warning] min_sum_hessian_in_leaf is set=11,
min_child_weight=0.001 will be ignored. Current value:
min_sum_hessian_in_leaf=11
[LightGBM] [Warning] min_data_in_leaf is set=6, min_child_samples=20
will be ignored. Current value: min_data_in_leaf=6
[LightGBM] [Warning] bagging_freq is set=5, subsample_freq=0 will be
ignored. Current value: bagging_freq=5
[LightGBM] [Warning] bagging_fraction is set=0.8, subsample=1.0 will
be ignored. Current value: bagging_fraction=0.8
[LightGBM] [Warning] feature_fraction is set=0.2319,
colsample_bytree=1.0 will be ignored. Current value:
feature_fraction=0.2319
[LightGBM] [Warning] min_sum_hessian_in_leaf is set=11,
min_child_weight=0.001 will be ignored. Current value:
min_sum_hessian_in_leaf=11
[LightGBM] [Warning] min_data_in_leaf is set=6, min_child_samples=20
will be ignored. Current value: min_data_in_leaf=6
[LightGBM] [Warning] bagging_freq is set=5, subsample_freq=0 will be
ignored. Current value: bagging_freq=5
[LightGBM] [Warning] bagging_fraction is set=0.8, subsample=1.0 will
be ignored. Current value: bagging_fraction=0.8
[LightGBM] [Warning] feature_fraction is set=0.2319,
colsample_bytree=1.0 will be ignored. Current value:
```

```
feature_fraction=0.2319
[LightGBM] [Warning] min_sum_hessian_in_leaf is set=11,
min_child_weight=0.001 will be ignored. Current value:
min_sum_hessian_in_leaf=11
[LightGBM] [Warning] min_data_in_leaf is set=6, min_child_samples=20
will be ignored. Current value: min_data_in_leaf=6
[LightGBM] [Warning] bagging_freq is set=5, subsample_freq=0 will be
ignored. Current value: bagging_freq=5
[LightGBM] [Warning] bagging_fraction is set=0.8, subsample=1.0 will
be ignored. Current value: bagging_fraction=0.8
[LightGBM] [Warning] feature_fraction is set=0.2319,
colsample_bytree=1.0 will be ignored. Current value:
feature_fraction=0.2319
[LightGBM] [Warning] min_sum_hessian_in_leaf is set=11,
min_child_weight=0.001 will be ignored. Current value:
min_sum_hessian_in_leaf=11
[LightGBM] [Warning] min_data_in_leaf is set=6, min_child_samples=20
will be ignored. Current value: min_data_in_leaf=6
[LightGBM] [Warning] bagging_freq is set=5, subsample_freq=0 will be
ignored. Current value: bagging_freq=5
[LightGBM] [Warning] bagging_fraction is set=0.8, subsample=1.0 will
be ignored. Current value: bagging_fraction=0.8
[LightGBM] [Warning] feature_fraction is set=0.2319,
colsample_bytree=1.0 will be ignored. Current value:
feature_fraction=0.2319
[LightGBM] [Warning] min_sum_hessian_in_leaf is set=11,
min_child_weight=0.001 will be ignored. Current value:
min_sum_hessian_in_leaf=11
[LightGBM] [Warning] min_data_in_leaf is set=6, min_child_samples=20
will be ignored. Current value: min_data_in_leaf=6
[LightGBM] [Warning] bagging_freq is set=5, subsample_freq=0 will be
ignored. Current value: bagging_freq=5
[LightGBM] [Warning] bagging_fraction is set=0.8, subsample=1.0 will
be ignored. Current value: bagging_fraction=0.8
LGBM score: 0.1167 (0.0059)
```

```
#Πάμε  τώρα να διαμορφώσουμε ένα μέσο αποτέλεσμα των 6 μοντέλων που
αναπτύχθηκαν παραπάνω

class AveragingModels(BaseEstimator, RegressorMixin,
TransformerMixin):
    def __init__(self, models):
        self.models = models

    # Ορίζουμε κλώνους των αρχικών μοντέλων για την προσαρμογή των
δεδομένων
    def fit(self, X, y):
        self.models_ = [clone(x) for x in self.models]
```

```
        # Train cloned base models
        for model in self.models_:
            model.fit(X, y)

        return self

    #Τώρα κάνουμε τις προβλέψεις για τα κλωνοποιημένα μοντέλα και τις
υπολογίζουμε κατά μέσο όρο
    def predict(self, X):
        predictions = np.column_stack([
            model.predict(X) for model in self.models_
        ])
        return np.mean(predictions, axis=1)

averaged_models = AveragingModels(models = (ENet, GBoost, KRR, lasso))

score = rmsle_cv(averaged_models)
print(" Averaged base models score: {:.4f} ({:.4f})\
n".format(score.mean(), score.std()))

 Averaged base models score: 0.1132 (0.0084)


class StackingAveragedModels(BaseEstimator, RegressorMixin,
TransformerMixin):
    def __init__(self, base_models, meta_model, n_folds=5):
        self.base_models = base_models
        self.meta_model = meta_model
        self.n_folds = n_folds

    # Προσαρμόζουμε και πάλι τα δεδομένα σε κλώνους των αρχικών
μοντέλων
    def fit(self, X, y):
        self.base_models_ = [list() for x in self.base_models]
        self.meta_model_ = clone(self.meta_model)
        kfold = KFold(n_splits=self.n_folds, shuffle=True,
random_state=156)

        # Εκπαιδεύουμε τα κλωνοποιημένα βασικά μοντέλα και στη
συνέχεια δημιουργούμε out-of-fold προβλέψεις
        # που χρειάζονται για την εκπαίδευση του κλωνοποιημένου μετα-
μοντέλου
        out_of_fold_predictions = np.zeros((X.shape[0],
len(self.base_models)))
        for i, model in enumerate(self.base_models):
            for train_index, holdout_index in kfold.split(X, y):
                instance = clone(model)
                self.base_models_[i].append(instance)
                instance.fit(X[train_index], y[train_index])
                y_pred = instance.predict(X[holdout_index])
```

```
                out_of_fold_predictions[holdout_index, i] = y_pred

        # Τώρα εκπαιδεύουμε το κλωνοποιημένο μετα-μοντέλο
χρησιμοποιώντας τις out-of-fold προβλέψεις ως νέο χαρακτηριστικό
        self.meta_model_.fit(out_of_fold_predictions, y)
        return self

    #Πραγματοποιούμε τις προβλέψεις όλων των βασικών μοντέλων στα
δεδομένα δοκιμής και χρησιμοποιούμε τις μέσες προβλέψεις ως
    #μετα-χαρακτηριστικά για την τελική πρόβλεψη που γίνεται από το
μετα-μοντέλο
    def predict(self, X):
    def predict(self, X):
        meta_features = np.column_stack([
            np.column_stack([model.predict(X) for model in
base_models]).mean(axis=1)
            for base_models in self.base_models_ ])
        return self.meta_model_.predict(meta_features)

stacked_averaged_models = StackingAveragedModels(base_models = (ENet,
GBoost, KRR),
                                                 meta_model = lasso)

score = rmsle_cv(stacked_averaged_models)
print("Stacking Averaged models score: {:.4f}
({:.4f})".format(score.mean(), score.std()))

Stacking Averaged models score: 0.1088 (0.0076)

def rmsle(y, y_pred):
    return np.sqrt(mean_squared_error(y, y_pred))

stacked_averaged_models.fit(endtrain.values, y_train)
stacked_train_pred = stacked_averaged_models.predict(endtrain.values)
stacked_pred =
np.expm1(stacked_averaged_models.predict(endtest.values))
print(rmsle(y_train, stacked_train_pred))

0.07625048017878562

model_xgb.fit(endtrain, y_train)
xgb_train_pred = model_xgb.predict(endtrain)
xgb_pred = np.expm1(model_xgb.predict(endtest))
print(rmsle(y_train, xgb_train_pred))

[05:28:34] WARNING: ../src/learner.cc:627:
Parameters: { "silent" } might not be used.

  This could be a false alarm, with some parameters getting used by
language bindings but
  then being mistakenly passed down to XGBoost core, or some parameter
```

```
actually being used
  but getting flagged wrongly here. Please open an issue if you find
any such cases.


0.07920860715942668
```

```
model_lgb.fit(endtrain, y_train)
lgb_train_pred = model_lgb.predict(endtrain)
lgb_pred = np.expm1(model_lgb.predict(endtest.values))
print(rmsle(y_train, lgb_train_pred))
```

```
[LightGBM] [Warning] feature_fraction is set=0.2319,
colsample_bytree=1.0 will be ignored. Current value:
feature_fraction=0.2319
[LightGBM] [Warning] min_sum_hessian_in_leaf is set=11,
min_child_weight=0.001 will be ignored. Current value:
min_sum_hessian_in_leaf=11
[LightGBM] [Warning] min_data_in_leaf is set=6, min_child_samples=20
will be ignored. Current value: min_data_in_leaf=6
[LightGBM] [Warning] bagging_freq is set=5, subsample_freq=0 will be
ignored. Current value: bagging_freq=5
[LightGBM] [Warning] bagging_fraction is set=0.8, subsample=1.0 will
be ignored. Current value: bagging_fraction=0.8
0.07169383068991829
```

```
'''RMSE on the entire Train data when averaging'''
```

```
print('RMSLE score on train data:')
print(rmsle(y_train,stacked_train_pred*0.70 +
               xgb_train_pred*0.15 + lgb_train_pred*0.15 ))
```

```
RMSLE score on train data:
0.07374341697367778
```

```
ensemble = stacked_pred*0.70 + xgb_pred*0.15 + lgb_pred*0.15
ensemble
```

```
array([120410.24474751, 157093.8448176 , 187140.72706248, ...,
       167030.19021957, 118801.58936055, 217893.77827308])
```

```
sub = pd.DataFrame()
sub['ID'] = test_data_ID
sub['SALEPRICE'] = ensemble
sub.to_csv('SUB_DIPLOMA_THESIS_ENSEMBLE.csv',index=False)
```