ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφοριακά Συστήματα & Υπηρεσίες»

Ειδίκευση: Μεγάλα Δεδομένα και Αναλυτική

Μάθημα: Η Γλώσσα Προγραμματισμού Python

Ονοματεπώνυμο	Αρ. Μητρώου
Γαρδέλης Σταύρος	**ΜΕ 1922**
Καπότης Χρήστος	**ΜΕ 1926**
Πλατής Κώστας	**ΜΕ 1940**
Τζαβάρας Γιάννης	**ΜΕ 1946**

Πίνακα περιεχομένων

Περιγραφή του προβλήματος

Η αγορά ενός σπιτιού είναι συνήθως η μεγαλύτερη αγορά που κάνει ο άνθρωπος σε όλη τη διάρκεια της ζωής του. Μια τέτοιου είδους κίνηση χρειάζεται πολύ σκέψη , καταγραφή και ανάλυση των αναγκών του κάθε ανθρώπου. Κάθε άνθρωπος είναι διαφορετικός και έτσι οι ανάγκες του αλλάζουν. Άρα πριν την αγορά μιας κατοικίας θα πρέπει πρώτα να έχει αξιολογηθεί το σύνολο αυτών των αναγκών για να προκύψει και το πιο σημαντικό κομμάτι , αυτό της τιμής της αγοράς. Σίγουρα παίζει ρόλο και η οικονομική κατάσταση της αγοράς της κάθε εποχής, αλλά κάθε ακίνητο αναλόγως με αυτά που προσφέρει (περιοχή, αριθμός υπνοδωματίων, θέρμανση, τετραγωνικά μέτρα, πάρκινγκ, αποθήκη κ.α.), έχει και διαφορετική τιμή.

Η μηχανική μάθηση (Machine Learning) μας παρέχει εργαλεία και αλγορίθμους ώστε να προβλέψουμε την τιμή αγοράς ενός ακινήτου αν έχουμε μια σειρά χαρακτηριστικών του ακινήτου. Κάποια χαρακτηριστικά έχουν μεγάλο βάρος και επηρεάζουν την τιμή πώλησης, κάποια άλλα πάλι όχι.

Στη συγκεκριμένη εργασία θα πραγματοποιηθεί πρόβλεψη της τιμής πώλησης κατοικιών της πόλης Ames στην Αϊόβα των Ηνωμένων πολιτειών. Τα δεδομένα τα οποία θα χρησιμοποιήσουμε βρίσκονται στο παρακάτω link: https://www.kaggle.com/c/house-prices-advanced-regression-techniques . Μέσα από δύο υποσύνολα με 80 & 81 χαρακτηριστικά σπιτιών θα πραγματοποιηθεί η εκπαίδευση του αλγόριθμού μας ώστε να μπορέσουμε να κάνουμε την πρόβλεψή μας σχετικά με την τιμή πώλησης της εκάστοτε κατοικίας. Πιο συγκεκριμένα το 1^{ο} υποσύνολο περιέχει ένα παραπάνω χαρακτηριστικό, αυτό της τιμής πώλησης , με το οποία θα γίνει η εκπαίδευση (train) του μοντέλου μας και το 2^{ο} υποσύνολο, το οποίο δεν περιέχει το συγκεκριμένο πεδίο και θα πρέπει να κάνουμε εμείς την πρόβλεψη της τελικής τιμής.

Περιγραφή δεδομένων

Το συγκεκριμένο σύνολο δεδομένων, περιέχει 2919 εγγραφές σπιτιών μαζί με ένα μεγάλο αριθμό επεξηγηματικών χαρακτηριστικών μεταβλητών. Το δείγμα ανήκει σε αγοραπωλησίες που πραγματοποιήθηκαν μεταξύ 2006 και 2010 για κατοικίες που χτίστηκαν μέχρι και «2 αιώνες» πριν και πιο συγκεκριμένα το 1872.

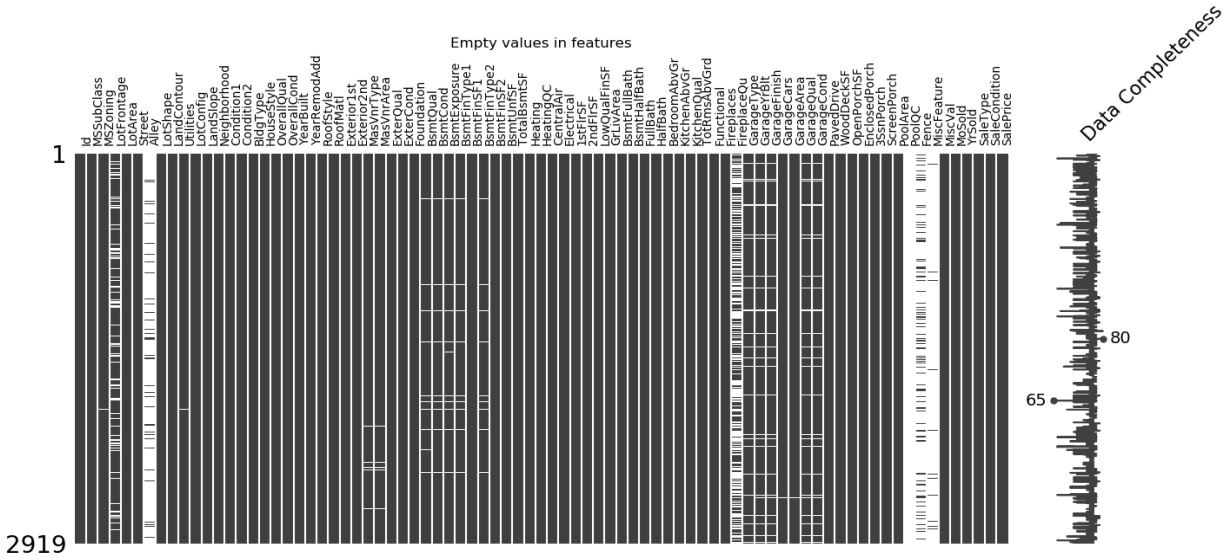
Όπως αναφέρθηκε και παραπάνω αυτές οι 2919 εγγραφές έχουν χωριστεί στη μέση σε 2 υποσύνολο με τη μόνη διαφορά τους να είναι ότι στο 2^{ο} λείπει η τιμή πώλησης του ακινήτου. Πάνω σε αυτό το υποσύνολο καλούμαστε να κάνουμε πρόβλεψη της τιμής πώλησης.

Οι μεταβλητές επικεντρώνονται ουσιαστικά στην ποιότητα και την ποσότητα πολλών φυσικών χαρακτηριστικών ενός ακινήτου. Οι περισσότερες αφορούν το είδος των πληροφοριών που ζητά ένας τυπικός αγοραστής να μάθει πριν αγοράσει μια κατοικία. Μερικά παραδείγματα είναι το έτος κατασκευής, το μέγεθος σε τ.μ., αριθμός υπνοδωματίων, μπάνια κα. Όλα αυτά τα χαρακτηριστικά συμμετέχουν στην αξιολόγηση μιας κατοικίας και αν θέλουμε να τα χωρίσουμε είναι 37 αριθμητικά και από 39 κατηγορικά χαρακτηριστικά.

Data Quality

Υπάρχουν πολλές προσεγγίσεις και μετρικές με τις οποίες μπορούμε να χρησιμοποιήσουμε για να χαρακτηρίσουμε ένα σύνολο δεδομένων ως προς την ποιότητα του. Ενδεικτικά κάποιοι από τους τρόπους με τους οποίους μπορεί κάποιος να αξιολογήσει την ποιότητα ενός dataset είναι το μέγεθος του dataset, η συνοχή των τιμών (μεγάλα – μικρά διαστήματα μεταξύ συμπληρωμένων τιμών), το σύνολο των κενών τιμών(ελλιπής τιμές), τον χρόνο συλλογής των δεδομένων (up to date data), την σχετικότητα των δεδομένων με την εργασία μας και άλλες παρόμοιες μετρικές.

Για την αξιολόγηση της ποιότητας του set “House Prices: Advanced Regression Techniques” ελέγξαμε το σύνολο των ελλειπουσών τιμών καθώς και των μοναδικών τιμών. Για τον έλεγχο χρησιμοποιήσαμε την μέθοδο *.info()* της βιβλιοθήκης *Pandas* καθώς και την βιβλιοθήκη missingno για τον σχεδιασμό του παρακάτω bar plot το οποίο απεικονίζει τη συνοχή και το πλήθος των ολοκληρωμένων *features*.



Επιπλέον, η μέθοδος *.info()* του *Pandas* μας επιστρέφει πληροφορίες σχετικά με τον τύπο, το πλήθος των nan τιμών και το μέγεθος του set. Για τον καθορισμό των nan values δημιουργήσαμε των πίνακα missing_values_set, στο οποίο περιλαμβάνουμε όλες εκείνες τις τιμές που θέλουμε η μέθοδος *.read_csv()* να ορίσει ως nan κατά την εισαγωγή του set.

Προ-επεξεργασία Δεδομένων

Για την προ εξεργασία των δεδομένων (αφού έγινε η ανάλυση των χαρακτηριστικών καθώς και η ένωση του train.csv και του test.csv με την μέθοδο *.concat()* από την βιβλιοθήκη *Pandas*) , διαγράφουμε τις στήλες (features) οι οποίες έχουν περισσότερες από 600 nan values. Με τον παρακάτω κώδικα επιλέξαμε και διαγράψαμε εκείνες τις στήλες που περιέχουν πάνω από 600 Nan values.

```
###### search for empty values in set
many_empty = [  attribute for attribute
                in full_dataset if
                full_dataset[attribute].isnull().sum() > 600     ]

##['Alley', 'FireplaceQu', 'PoolQC', 'Fence', 'MiscFeature']

full_dataset = full_dataset.drop(columns=many_empty)
#### drop columns with over of 600 empty values
```

Για την αποκατάσταση των υπολοίπων τιμών που λείπουν από τα δεδομένα μας λειτουργήσαμε ώς εξής :

- Αριθμητικά δεδομένα: Ανάθεση της τιμής *.median()* (διάμεσος) της αντίστοιχης στήλης.
- Κατηγορικά δεδομένα: Ανάθεση της τιμής που εμφανίζετε περισσότερες φορές στην αντίστοιχη στήλη.

Η παραπάνω διαδικασία υλοποιήθηκε με μία for loop καθώς και των δημιουργία δύο πινάκων numerical_attr (αριθμητικές στήλες του set) και categorical_attr (στήλες που περιλαμβάνουν strings).

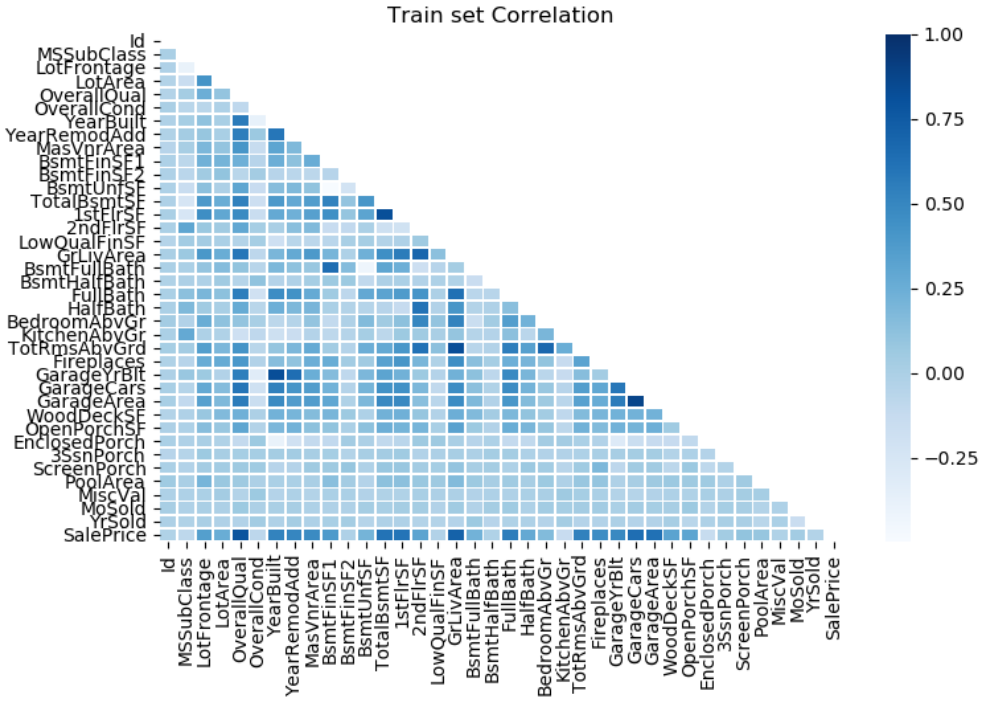
```
#### fill missing values  numerical ---> np.mean    , categorigal--->  most
frequent value
for attributes in full_dataset:
    if attributes in numerical_attr:
        full_dataset[attributes] =
full_dataset[attributes].fillna(value=full_dataset[attributes].median())
    else:
        full_dataset[attributes]=
full_dataset[attributes].fillna(value=full_dataset[attributes]
.value_counts().idxmax() )
```

Στο παραπάνω κομμάτι κώδικα χρησιμοποιήθηκαν επίσης οι μέθοδοι :

- *.fillna() :*
- *.value_counts()*
- *.idxmax()*

Συσχετίσεις Χαρακτηριστικών

Για την ανακάλυψη συσχετίσεων μεταξύ των χαρακτηριστικών μας δημιουργήσαμε ένα (τριγωνικό) heatmap του train_set (με το χαρακτηριστικό 'SalePrice').



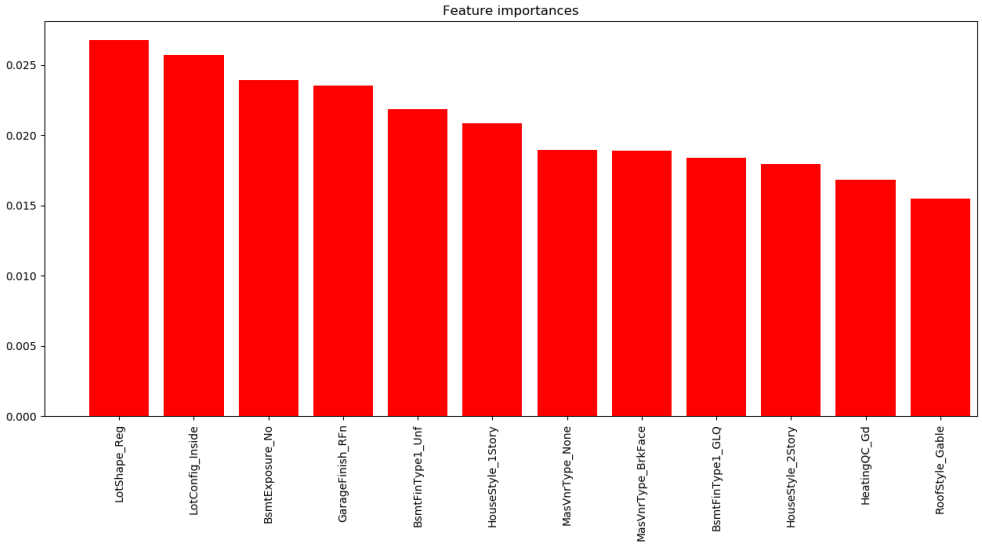
Παρατηρούμε την έντονη συσχέτιση των χαρακτηριστικών (σκούρο μπλε) OverallQuall, TotalBsmtSF, GrLivArea με το SalePrice.

Η συσχέτιση μεταξύ των δεδομένων του set μας δίνεται από την μέθοδο *.corr()* του *Pandas*.

Αξιολόγηση Κατηγορικών Χαρακτηριστικών

Για την μετατροπή των κατηγορικών δεδομένων σε αριθμητικά χρησιμοποιήθηκε η μέθοδος *get_dummies()* του *Pandas*. Η μετατροπή όλων των κατηγορικών δεδομένων σε αριθμητικά θα οδηγήσει σε ένα πρόβλημα με εκατοντάδες χαρακτηριστικά (πολλές διαστάσεις). Για τον λόγο αυτό επιλέξαμε να μετατρέψουμε κάποια από τα κατηγορικά χαρακτηριστικά. Η επιλογή αυτή έγινε με την βοήθεια ενός Random Forest κατηγοριοποιητή. Πιο συγκεκριμένα, χρησιμοποιήθηκε η κλάση *SelectFromModel* της βιβλιοθήκης *sklearn* και ο αλγόριθμος *RandomForestClassifier* για την επιλογή των χαρακτηριστικών με το μεγαλύτερο βάρος. Αφού μετατρέψαμε όλα τα κατηγορικά χαρακτηριστικά σε αριθμητικά με την μέθοδο *get_dummies* στην συνέχεια κάναμε fit τον *RandomForestClassifier* με να αξιολογήσουμε την σημαντικότητα κάθε χαρακτηριστικού. Η εφαρμογή του αλγορίθμου έγινε μόνο στο αρχικό μας train_Set (στην βιβλιογραφία αναφέρετε ως καλή τεχνική αποφυγής του overfitting). Ως τιμή *threshold* της μεθόδου *SelectFromModel* χρησιμοποιήθηκε η μέση τιμή του importance όλων των κατηγορικών χαρακτηριστικών. Βρίσκοντας το importance των

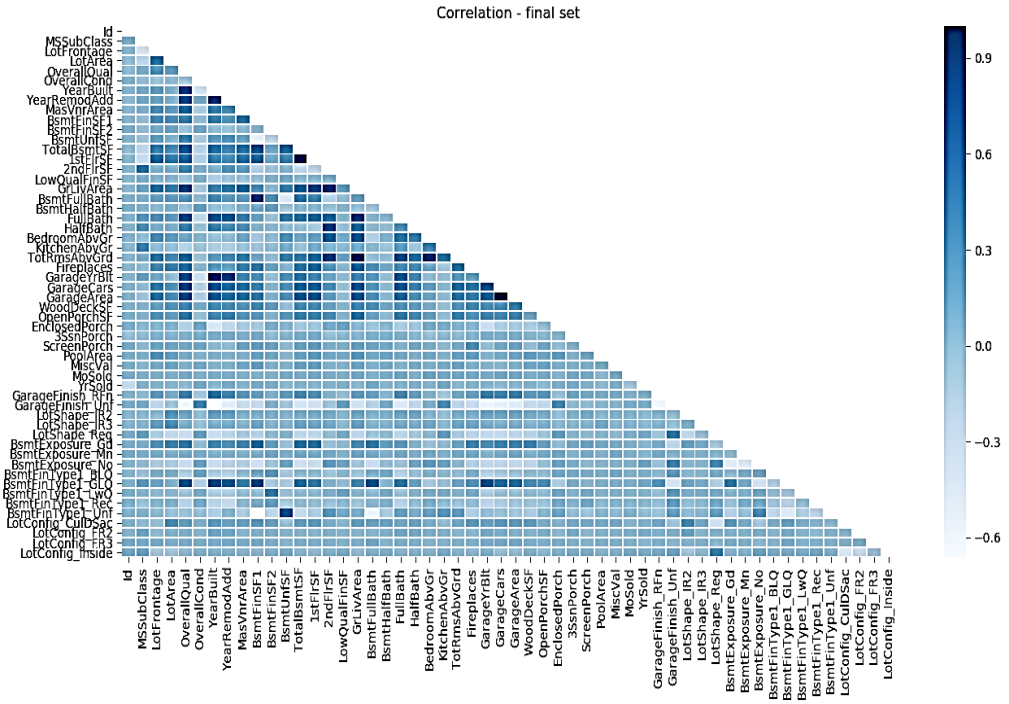
χαρακτηριστικών, επιλέγουμε να μετατρέψουμε τα χαρακτηριστικά *"GarageFinish" ,"LotShape", "BsmtExposure", "BsmtFinType1","LotConfig".*



Κατατάξεις και Σχολιασμοί

Για την δημιουργία των Κατατάξεων (κατατάξεις : A,B,C,D,E,F, ζητούμενα εργασίας) χρησιμοποιήθηκε το set δεδομένων το οποίο περιείχε όλα τα αριθμητικά features και τα 5 κατηγορικά features που επιλέξαμε και μετατρέψαμε παραπάνω.

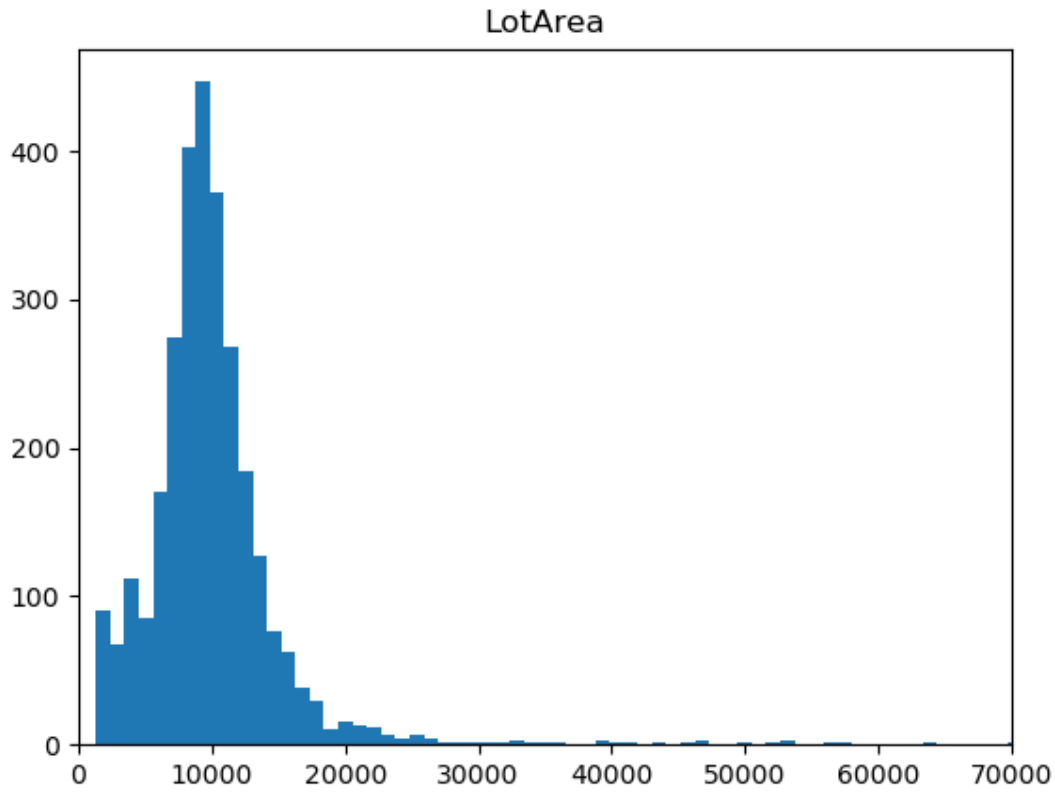
- **Κατάταξη Α:**



Ο αθροιστικός συντελεστής συσχέτισης ήταν απόκριση του αθροίσματος κάθε σειράς του pd.Dataframe που δημιουργήθηκε από τις σχετίσεις του set (*.corr()*). Το GrLivArea είναι το χαρακτηριστικό με το μεγαλύτερο αθροιστικό συντελεστή δηλαδή έχει μεγάλη συσχέτιση με όλα τα υπόλοιπα χαρακτηριστικά. Μεγάλες συσχετίσεις μεταξύ των χαρακτηριστικών μας δεν είναι απαραίτητα καλό μιας και μπορεί να οδηγήσει σε *Multicollinearity Problem* το οποίο έχει μεγάλη επίδραση στην γραμμική Παλινδρόμηση.

- **Κατάταξη Β:**

Για την κατάταξη Β υπολογίσαμε με την βοήθεια της βιβλιοθήκης *statistics* υπολογίσαμε την μέση τιμή, την τυπική απόκλιση και την διακύμανση (*mean(), stdev(), variance()*). Στην συνέχεια τα στοιχεία αυτά αποθηκευτήκαν σε ένα *pd.Dataframe* και ταξινομήθηκαν ως προς την τυπική απόκλιση. Το χαρακτηριστικό *LotArea* παρουσιάζει την μεγαλύτερη τυπική απόκλιση.



- **Κατάταξη C:**

Με την χρήση της βιβλιοθήκης *scipy.stats* και της μεθόδου *zscore* κανονικοποιήσαμε τα δεδομένα του set μας. Στην συνέχεια ακολουθήσαμε την ίδια διαδικασία όπως και για την κατάταξη Α. Παρατηρούμε ότι τα αποτελέσματα είναι ίδια με αυτά της κατάταξης Α. Δηλαδή ο αθροιστικός συντελεστής συσχέτισης κάθε μεταβλητής **δεν επηρεάστηκε** από την κανονικοποίηση.

- **Κατάταξη D:**

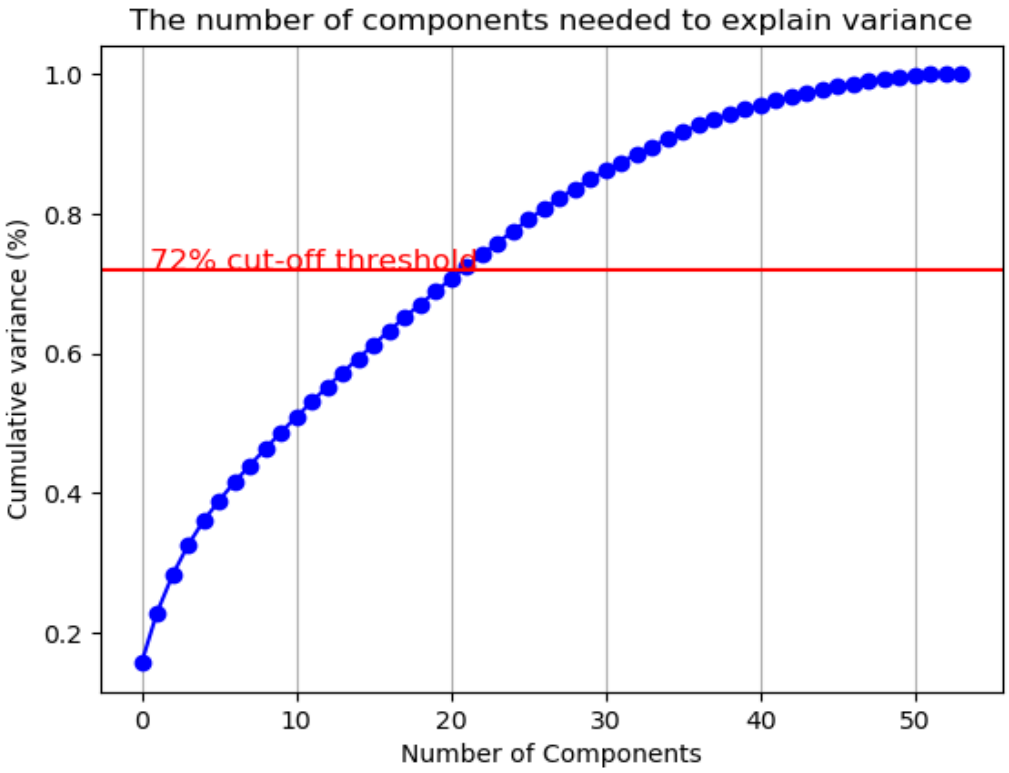
  Για την κατάταξη D ακολουθήσαμε την ίδια διαδικασία όπως και στην κατάταξη B με την μόνη διαφορά ότι τώρα χρησιμοποιήσαμε το κανονικοποιημένο Set μας. Η κανονικοποίηση είχε ως αποτέλεσμα όλα τα *Features* να αποχτήσουν **την ίδια** τυπική απόκλιση.

- **Κατατάξεις E και F :**

  Με χρήση της μεθόδου PCA της βιβλιοθήκης *sklearn.decomposition* εφαρμόσαμε PCA στα 2 set (κανονικοποιημένο και μη).

PCA

Για την επιλογή του αριθμού των components που θέλουμε από το PCA δημιουργήσαμε ένα διάγραμμα με την αθροιστική διακύμανση των n_components. Επιλέγουμε 20 components. Τα 20 αυτά Components $P_1, \dots, P_{20}$ περιέχουν το 72% της συνολικής πληροφορίας του αρχικού κανονικοποιημένου μας set κατάταξη E.



Η επιλογή περισσότερων components για την αναπαράσταση των χαρακτηριστικών μας μπορεί να μοιάζει ότι δίνει καλύτερα αποτελέσματα (περισσότερη πληροφορία), δεν είναι όμως αναγκαστικά καλό μιας και εκτός από περισσότερη πληροφορία «κερδίζουμε» και περισσότερες εσφαλμένες τιμές (πχ. Θόρυβος).

Για την υλοποίηση του plot χρησιμοποιήθηκε η μέθοδος *.cumsum()* της βιβλιοθήκης NumPy όπου προσθέτει κάθε φορά το variance πληροφορίας που παίρνουμε από κάθε επόμενο component.

```
pca_search = PCA().fit(full_dataset[numerical_attr])
# print(pca_search.explained_variance_ratio_)
fig, ax = plt.subplots()
print("PCA EXPLAINED VAR RATIO",pca_search.explained_variance_ratio_)
y = np.cumsum(pca_search.explained_variance_ratio_)          #get variance ratio
cumsum for generate the plot  #cumsum[1,2,3]=[1,3,6] etc.
xi = np.arange(0, len(y), step=1)                                  #number of
features
plt.plot(xi,y,marker='o', color='b')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative variance (%)')
plt.title('The number of components needed to explain variance')
```

Με την χρήση της μεθόδου *.explained_variance_ratio_* μπορούμε να διαπιστώσουμε το ποσοστό της διακύμανσης που εξηγείται από κάθε component. Συγκεκριμένα έχουμε:

#	EXPL_RATIO
P1	.15947563
P2	.07355939
P3	.05456572
P4	.04379911
P5	.03542295
P6	.0293377
P7	.0262918
P8	.02430604
P9	.02396595
P10	.0226734
P11	.02255183
P12	.02213547
P13	.02149326
P14	.02065379
P15	.02039004
P16	.02012774
P17	.01999716
P18	.01914499
P19	.0188401
P20	.01865711

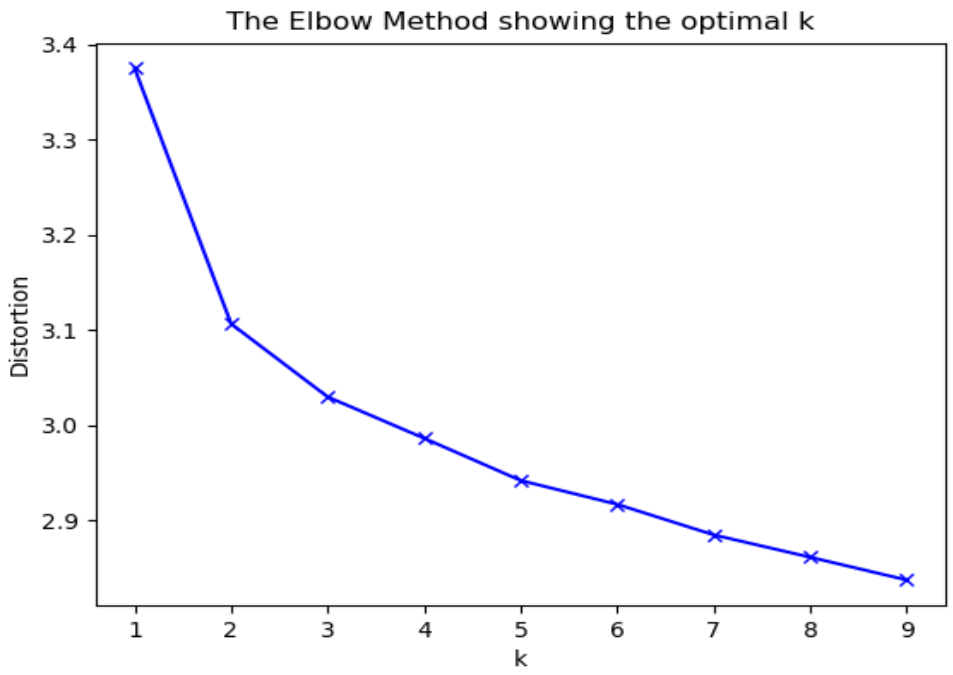
Επιπλέον με την μέθοδο .components_ μπορούμε να δούμε την συμβολή του κάθε feature σε κάθε ένα component (σε μία κλίμακα από -1 έως 1). Για παράδειγμα, για τα 5 πρώτα components έχουμε:

	MSSubClass	LotFrontage	...	LotConfig_FR3	LotConfig_Inside
0	-0.013240	0.134833	...	0.010581	-0.038032
1	0.087256	0.020944	...	-0.018224	0.000871
2	-0.184106	0.258273	...	-0.018331	-0.083372
3	0.348852	-0.187096	...	0.023999	-0.000028
4	-0.124793	-0.013576	...	0.037773	-0.449477

[5 rows x 53 columns]

Clustering

Για την επιλογή του αριθμού των cluster που θα επιλέξουμε για την εκπαίδευση του *Kmeans* χρησιμοποιήθηκε η μέθοδος «Elbow».



Στο παραπάνω γράφημα στον άξονα-Χ έχουμε το πλήθος των clusters και στον άξονα-Υ το μέτρο *Distortion* ή αλλιως *sum of square errors (SSE).* Επιλέγουμε πλήθος clusters= 2.

Από την εκπαίδευση του *Kmeans* προκύπτουν 2 clusters.

#	Total length	Empty SalePrice
Cluster_0	1381	674
Cluster_1	1538	785

Το άθροισμα των τετραγώνων των αποστάσεων από τα centroids είναι 12195.155

Στην συνέχεια εφαρμόσαμε Linear Regression κάνοντας fit και predict σε κάθε ένα cluster ξεχωριστά. Μεταχειριζόμαστε δηλαδή κάθε ένα cluster σαν dataset.

Πιο αναλυτικά, διαχωρίσαμε το *pd.Dataframe* με την βοήθεια των μεθόδων *labels* και *predict* ώστε να διαχωρίσουμε τις εγγραφές στις δύο κλάσεις.

```
reate dataframes cluster_0 (all data in cluster 0 ) & cluster_1 ( all data in
cluster 1 )
cluster_0 = pd.DataFrame(  data= pca_data[ pca_data['Cluster'] == 0 ] ,
columns=[..])
cluster_1 = pd.DataFrame(  data= pca_data[ pca_data['Cluster'] == 1 ] ,
columns=[..]            )
```

Στην συνέχεια για κάθε ένα από τα δύο set δημιουργήθηκε ένα train_set και ένα test_set με βάση το column '*SalePrice*'

```
for i in range(0,2,1):          #create sets , train, test
        train = []
        test = []
        for row in clusters[i].values:
                x = row[22]          # SalePrice column
                if x == " ":                  #if is empty
                        test.append(row)        #append to test set
                else:
                        train.append(row)          # append to train set
```

Δημιουργώντας το μοντέλο της γραμμικής παλινδρόμησης από την βιβλιοθήκη *sklearn.linear_model* κάνουμε *Fit* (στα Xtrain ,Ytrain) και στη συνέχεια *Predict* (στα Xtest δεδομένα μας). Για την αποθήκευση των αποτελεσμάτων δημιουργούνται δύο πίνακες results, id οπού περιέχουν τα αποτελέσματα των *predict* καθώς και τα id των εγγραφών. Τέλος, δημιουργείτε ένα Dataframe το οποίο περιέχει τις στήλες id και SalePrice. Με την βοήθεια της μεθόδου *pd.to_csv()* δημιουργούμε το CSV file για την υποβολή στο Kaggle.

Αποτελέσματα

Τα αποτελέσματα αξιολογήθηκαν από τον καθηγητή του μαθήματος , ο οποίος γνώριζε τις τελικές τιμές του SalePrice.

Όσο αναφορά τον σχολιασμό των χαρακτηριστικών παρατηρούμε ότι τα χαρακτηριστικά *KitchenAbvGr* (1.491591), *PoolArea* (1.039873), *MiscVal* (1.354781), *GarageFinish*_RFn (1.152697) φαίνεται να αποτελούν χαρακτηριστικά με μεγάλη βαρύτητα μιας και έχουν στην ανάλυση σε PC το μεγαλύτερο μέρος πληροφορίας. Στον αντίποδα βρίσκονται τα χαρακτηριστικά *LotConfig_Inside* (-1.315510), *OverallCond* (-0.958376) , *GarageFinish_Unf* (-0.773561).

Οι τιμές στις παρενθέσεις είναι το άθροισμα των διανυσμάτων μέγιστης διακύμανσης των χαρακτηριστικών για όλα τα components.

Επιπλέον έγινε σύγκριση αποτελεσμάτων με τα αποτελέσματα που μας έδωσε η συσταδοποίηση και η παλινδρόμηση που εφαρμόστηκε στα 20 πρώτα χαρακτηριστικά της Κατάταξης Α (αθροιστικός συντελεστής συσχέτισης που μια μεταβλητή παρουσιάζει με όλες τις υπόλοιπες μεταβλητές, σε φθίνουσα σειρά). Η συσταδοποίηση με αριθμό συστάδων 2 ομαδοποίησε τα δεδομένα σε δύο κλάσης μεγέθους 2915 και 4 εγγραφών. Επιπλέον, η παλινδρόμηση μας έδωσε RMSE = 0.49570 (μέτρηση Kaggle).

Βιβλιοθήκες Python

Για την υλοποίηση του κώδικα χρησιμοποιήθηκαν οι παρακάτω βιβλιοθήκες και μέθοδοι(κυριότερες):

- *Pandas*
  - *read_csv()*
  - *corr()*
  - *concat()*
  - *DataFrame()*
  - *isnull()*
  - *info()*
  - *drop()*
  - *select_dtypes()*
  - *fillna()*
  - *get_dummies()*
  - *sort_values()*
  - *to_csv()*
  - *value_counts()*
- *sklearn*
  - *ensemble.RandomForestClassifier()*
- *sklearn*
  - *cluster.Kmeans()*
  - *linear_model. LinearRegression()*
  - *model_selection.train_test_split()*
  - *decomposition.PCA()*
  - *feature_selection.SelectFromModel()*
- *NumPy*
  - *zeros_like()*
  - *argsort()*
- *statistics*
  - *stdev()*
  - *variance()*
  - *mean()*
- *scipy*
  - *stats.zscore*
  - *spatial.distance.cdist()*
- *matplotlib*
  - *pyplot*

Συμπεράσματα και Επεκτάσεις

Κατά την διαδικασία ανάπτυξης του κώδικα και επίλυσης της εργασίας χρησιμοποιήθηκαν και άλλες τεχνικές – προσεγγίσεις οι οποίες όμως δεν έδωσα καλύτερο RMSE. Κάποιες από αυτές ήταν:

- ✓ Η επιλογή διαφορετικού πλήθους *n_components* (5,10,14).
- ✓ Η μετατροπή περισσότερων(/λιγότερων) κατηγορικών χαρακτηριστικών σε αριθμητικά.
- ✓ Η δημιουργία Pipeline για την εφαρμογή μη γραμμικής παλινδρόμησης. Στην προσέγγιση αυτή έγινε χρήση των βιβλιοθηκών *sklearn.preprocessing ,sklearn.pipeline* και οι μέθοδοι *PolynomialFeatures, LassoCV, make_pipeline.*

Ο κώδικας σε μεγάλο μέρος του μπορεί να χαρακτηριστεί διανυσματικός, και στα περισσότερα σημεία παράγει αυτοματοποιημένα αποτελέσματα.

Άλλες προσεγγίσεις οι οποίες συζητήθηκαν σε θεωρητικό επίπεδο αλλά δεν υλοποιήθηκαν ήταν:

- ➢ Η εξερεύνηση όλου του σετ δεδομένων με Random Forest για την επιλογή features (και όχι μόνο των κατηγορικών χαρακτηριστικών).
- ➢ Η περαιτέρω μελέτη των cluster που δημιουργήθηκαν για την ανακάλυψη προτύπων(πχ. χαρακτηριστικών που είναι κοντά τα centroids, είδος συσχέτισης μεταξύ των χαρακτηριστικών σε κάθε cluster κλπ.)
- ➢ Εφαρμογή διαφορετικών αλγορίθμων παλινδρόμησης για την εξαγωγή των αποτελεσμάτων.