Ενότητα 6 – Supervised Learning

Linear Regression

Αθανάσιος Σάκκας

ΟΠΑ

Γραμμική Παλινδρόμηση

- Η γραμμική παλινδρόμηση είναι ένα πολύ δημοφιλές εργαλείο.
- Αν υποθέσουμε ότι το μοντέλο είναι γραμμικό, δεν απαιτείται τεράστιος όγκος δεδομένων.
- Στη μηχανική μάθηση:
  - ο σταθερός όρος ονομάζεται **bias**
  - οι συντελεστές ονομάζονται **weights**

Για n παρατηρήσεις και m χαρακτηριστικά:

$$Y = a + b_1 X_1 + b_2 X_2 + \cdots + b_m X_m + \varepsilon$$

Mean Squared Error

Η τυπική προσέγγιση είναι να επιλέξουμε το a και τα b_i ώστε να ελαχιστοποιείται το μέσο τετραγωνικό σφάλμα:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- Το MSE αξιολογεί την προγνωστική απόδοση.
- Συγκρίνουμε την απόδοση σε:
  - training data
  - test data
- Σε πιο γενικές περιπτώσεις μπορούμε να χρησιμοποιήσουμε **gradient descent**.

Αλγόριθμος Gradient Descent

Στόχος: ελαχιστοποίηση μιας αντικειμενικής συνάρτησης αλλάζοντας παραμέτρους.

Βήματα:

1. Επιλέγουμε αρχικές τιμές για τις παραμέτρους
2. Βρίσκουμε την κατεύθυνση της πιο απότομης κλίσης
3. Κάνουμε ένα βήμα προς τη σωστή κατεύθυνση
4. Επαναλαμβάνουμε
5. Σταματάμε όταν υπάρξει σύγκλιση

Overfitting και Variable Selection

- Περισσότερες μεταβλητές δεν σημαίνουν πάντα καλύτερο μοντέλο.
- Η προσθήκη πολλών χαρακτηριστικών μπορεί να οδηγήσει σε **overfitting**.
- Τα overfitted μοντέλα περιγράφουν τυχαίο θόρυβο και όχι την πραγματική σχέση.
- Συνήθως έχουν χειρότερη out-of-sample πρόβλεψη.

Στόχος:

- ένα **parsimonious** μοντέλο
- με καλή προγνωστική απόδοση

Correlation Problem

Έστω δύο πολύ συσχετισμένα χαρακτηριστικά για την τιμή κατοικίας:

$$X_1 = \text{τετραγωνικά υπογείου}, \qquad X_2 = \text{τετραγωνικά πρώτου ορόφου}$$

Ένα overfitted μοντέλο μπορεί να είναι:

$$Price = 10X_1 - 9.5X_2$$

Ένα πιο απλό και σταθερό μοντέλο μπορεί να είναι:

$$Price = 0.5X_1 \qquad \text{ή} \qquad Price = 0.5X_2$$

Regularization

Το regularization είναι ένας τρόπος:

- αποφυγής του overfitting
- απλοποίησης του μοντέλου
- αντιμετώπισης της πολυσυγγραμμικότητας

Κύριες μέθοδοι:

- Ridge
- Lasso
- Elastic Net

Σημαντικό: πριν την εφαρμογή regularization, συνήθως κάνουμε scaling των χαρακτηριστικών.

Γενική Ιδέα του Regularization

Αντί να ελαχιστοποιούμε μόνο το σφάλμα προσαρμογής, προσθέτουμε και μια ποινή στο μέγεθος των συντελεστών:

$$\min_{\beta_0,\beta} \left\{ \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta_0 - x_i'\beta)^2 + \text{Penalty}(\beta) \right\}$$

Άρα:

- μικρότεροι συντελεστές
- μικρότερη πολυπλοκότητα
- καλύτερη γενίκευση σε νέα δεδομένα

Ridge Regression

Η Ridge regression λύνει:

$$\min_{\beta_0,\beta} \left\{ \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta_0 - x_i'\beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

- Χρησιμοποιεί **L2 penalty**
- Μειώνει το μέγεθος των συντελεστών
- Συνήθως δεν μηδενίζει ακριβώς συντελεστές
- Είναι χρήσιμη όταν οι predictors είναι πολύ συσχετισμένοι

Γιατί Ridge?

Η Ridge είναι χρήσιμη όταν:

- υπάρχει **multicollinearity**
- θέλουμε να κρατήσουμε όλες τις μεταβλητές στο μοντέλο
- θέλουμε πιο σταθερές εκτιμήσεις

Επίδραση του λ:

- μικρό $\lambda \Rightarrow$ μικρή συρρίκνωση
- μεγάλο $\lambda \Rightarrow$ ισχυρή συρρίκνωση προς το 0

Προσοχή: το παραδοσιακό inference δεν εφαρμόζεται άμεσα όπως στην OLS.

Lasso Regression

Η Lasso regression λύνει:

$$\min_{\beta_0,\beta} \left\{ \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta_0 - x_i'\beta)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

- Χρησιμοποιεί **L1 penalty**
- Κάνει shrinkage
- Μπορεί να θέσει κάποιους συντελεστές ακριβώς ίσους με 0
- Άρα κάνει και **variable selection**

Γιατί Lasso?

Η Lasso είναι χρήσιμη όταν:

- υπάρχουν πολλές μεταβλητές
- περιμένουμε ότι μόνο λίγες είναι πραγματικά σημαντικές
- θέλουμε πιο **sparse** και ερμηνεύσιμο μοντέλο

Κύρια ιδέα:

- μερικοί συντελεστές μειώνονται ακριβώς στο 0
- άρα η μέθοδος επιλέγει αυτόματα μεταβλητές

Elastic Net Regression

Η Elastic Net λύνει:

$$\min_{\beta_0,\beta} \left\{ \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta_0 - x_i'\beta)^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} \beta_j^2 \right\}$$

- Συνδυάζει Lasso και Ridge
- Κάνει shrinkage
- Μπορεί να κάνει variable selection
- Είναι χρήσιμη όταν υπάρχουν πολλές και συσχετισμένες μεταβλητές

Ridge vs Lasso vs Elastic Net

	Ridge	Lasso	Elastic Net
Penalty	L_2	L_1	$L_1 + L_2$
Shrinkage	Ναι	Ναι	Ναι
Variable selection	Όχι συνήθως	Ναι	Ναι
Correlated predictors	Πολύ καλό	Μερικές φορές ασταθές	Πολύ καλό
Interpretability	Μέτρια	Υψηλή	Υψηλή

Penalty Intuition

Ridge

Η ποινή $\sum \beta_j^2$ τιμωρεί περισσότερο τους μεγάλους συντελεστές, αλλά συνήθως δεν τους μηδενίζει.

Lasso

Η ποινή $\sum |\beta_j|$ ευνοεί λύσεις με ακριβώς μηδενικούς συντελεστές.

Elastic Net

Ο συνδυασμός L_1 και L_2 δίνει ταυτόχρονα σταθερότητα και επιλογή μεταβλητών.

Bias–Variance Tradeoff

Το regularization αλλάζει το tradeoff μεταξύ bias και variance:

- αυξάνει ελαφρώς το **bias**
- μειώνει τη **variance**
- συχνά βελτιώνει το **test error**

Άρα:

- λιγότερο ευαίσθητο μοντέλο σε μικρές αλλαγές των δεδομένων
- καλύτερη out-of-sample απόδοση

Επιλογή του Hyperparameter λ

Υπάρχουν διάφοροι τρόποι επιλογής του λ:

- Adjusted R^2
- AIC
- SBC / BIC
- **Cross-validation**

Στην πράξη, πολύ συχνά χρησιμοποιείται ***k*-fold cross-validation**.

k-Fold Cross Validation

Διαδικασία:

1. Χωρίζουμε τα training data σε k περίπου ίσα μέρη
2. Κρατάμε ένα fold για validation
3. Κάνουμε fit στα υπόλοιπα $k-1$ folds
4. Υπολογίζουμε το prediction error στο validation fold
5. Επαναλαμβάνουμε για όλα τα folds
6. Επιλέγουμε το λ που ελαχιστοποιεί το cross-validated MSE

$$\lambda^\star = \arg\min_\lambda CV(\lambda)$$

Εφαρμογές στα Χρηματοοικονομικά

Οι μέθοδοι regularization είναι πολύ χρήσιμες σε:

- πρόβλεψη αποδόσεων με πολλούς predictors
- factor selection
- επιλογή μακροοικονομικών μεταβλητών
- διαχείριση πολυσυγγραμμικότητας
- forecasting risk premia

Παραδείγματα predictors:

- valuation ratios
- momentum
- volatility
- macroeconomic indicators

Συμπέρασμα

- Η γραμμική παλινδρόμηση είναι απλή και ισχυρή μέθοδος.
- Το regularization βοηθά στην αποφυγή overfitting.
- Η **Ridge** είναι κατάλληλη για shrinkage και multicollinearity.
- Η **Lasso** είναι κατάλληλη για shrinkage και variable selection.
- Η **Elastic Net** συνδυάζει τα πλεονεκτήματα και των δύο.

Μικρότερη πολυπλοκότητα $\Rightarrow$ καλύτερη γενίκευση

Εφαρμογή Iowa House Price

Ανοίξτε το αρχείο Original_Data.xlsx. που περιλαμβάνει 80 χαρακτηριστικά για 2908 σπίτια στην Αϊόβα.

- Ο στόχος είναι να προβλέψουμε τις τιμές των σπιτιών στην Αϊόβα με βάση τα χαρακτηριστικά. Παρατηρείτε ότι υπάρχουν missing values (NA).
- Μετά από data cleaning καταλήγουμε σε 2908 σπίτια και θα χρησιμοποιήσουμε 47 χαρακτηριστικά.
- Χρησιμοποιούμε 1800 παρατηρήσεις στο training set, 600 στο validation set και 508 στο test set.

- Ανοίξτε το αρχείο *Houseprice_data_scaled.xlsx* που περιλαμβάνει τα δεδομένα που θα χρησιμοποιήσουμε στην ανάλυσή μας

❖ 21 numerical variables

❖2 categorical variables

Lot area (sq ft)	Number of half bathrooms
Overall quality (scale from 1 to 10)	Number of bedrooms
Overall condition (scale from 1 to 10)	Total rooms above grade
Year built	Number of fireplaces
Year remodeled	Parking spaces in garage
Basement finished sq ft	Garage area sq ft
Basement unfinished sq ft	Wood deck sq ft
Total basement sq ft	Open porch sq ft
1st floor sq ft	Enclosed porch sq ft
2nd floor sq ft	Neighborhood (25 alternatives)
Living area	Basement quality (6 natural ordering)
Number of full bathrooms	

Αποτελέσματα - Linear Regression

- Ανοίξτε το αρχείο python *6.1 linear_regression.ipynb*

```
]: #  MSE at the train set
   lr=LinearRegression()
   lr.fit(X_train,y_train)
   pred=lr.predict(X_train)
   print(mse(y_train,pred))

   0.1140152643124634
```

```
]: # MSE at the validation dataset
   lr.fit(X_train,y_train)
   pred=lr.predict(X_val)
   print(mse(y_val,pred))

   0.11702499460121657
```

```
# Create dataFrame with corresponding feature and its respective coefficients
coeffs = pd.DataFrame(
    [
        ['intercept'] + list(X_train.columns),
        list(lr.intercept_) + list(lr.coef_[0])
    ]
).transpose().set_index(0)

lreg_coefficient = pd.DataFrame()
lreg_coefficient["Features"] = X_train.columns
lreg_coefficient['Coef Estimate'] = pd.Series(lr.coef_[0])
print(lreg_coefficient)
```

Ορισμένα weights είναι αρνητικά όταν θα περιμέναμε να είναι θετικά.

Features	Weights	Features	Weights
LotArea	0.079	BrkSide	0.021
OverallQual	0.214	ClearCr	-0.007
OverallCond	0.096	CollgCr	-0.007
YearBuilt	0.161	Crawfor	0.036
YearRemodAdd	0.025	Edwards	-0.001
BsmtFinSF1	0.091	Gilbert	-0.008
BsmtUnfSF	-0.033	IDOTRR	-0.002
TotalBsmtSF	0.138	MeadowV	-0.016
1stFlrSF	0.153	Mitchel	-0.028
2ndFlrSF	0.133	Names	-0.039
GrLivArea	0.161	NoRidge	0.052
FullBath	-0.021	NPkVill	-0.022
HalfBath	0.017	NriddgHt	0.124
BedroomAbvGr	-0.084	NWAmes	-0.052
TotRmsAbvGrd	0.083	OLDTown	-0.026
Fireplaces	0.028	SWISU	-0.004
GarageCars	0.038	Sawyer	-0.018
GarageArea	0.052	SawyerW	-0.028
WoodDeckSF	0.021	Somerst	0.028
OpenPorchSF	0.034	StoneBr	0.063
EnclosedPorch	0.007	Timber	-0.003
Blmngtn	-0.018	Veenker	0.002
Blueste	-0.013	Bsmt Qual	0.011
BrDale	-0.025		

Αποτελέσματα - Ridge Regression

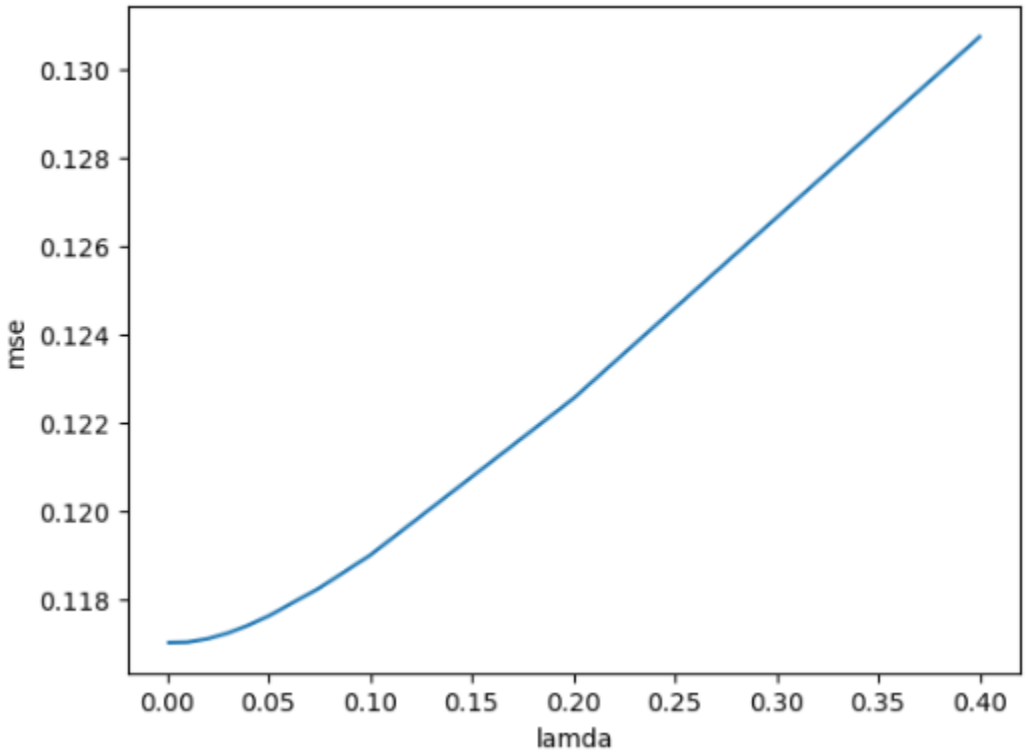
- Ανοίξτε το αρχείο python *6.2 ridge_regression.ipynb*
- Για το traiuset

```
# Importing Ridge
from sklearn.linear_model import Ridge

# The alpha used by Python's ridge should be the lambda times the number of observations
alphas=[0.001*1800,0.01*1800, 0.02*1800, 0.03*1800, 0.04*1800, 0.05*1800, 0.075*1800,0.1*1800,0.2*1800, 0.4*1800]
#alphas = np.linspace(0, 3.0, num=20)*1800 # from 0 to 3 with 20 numbers inside
#alphas = range(0,2*1800)
mses=[]
for alpha in alphas:
    ridge=Ridge(alpha=alpha)
    ridge.fit(X_train,y_train)
    pred=ridge.predict(X_val)
    mses.append(mse(y_val,pred))
    print(mse(y_val,pred))
```

```
0.11702368844694085
0.11703284346091346
0.11710797319753001
0.11723952924901127
0.11741457158889515
0.11762384068711462
0.11825709631198018
0.11900057469147929
0.12254649996292954
0.1307359968074713
```

```
plt.xlabel('lamda')
plt.ylabel('mse')
lamdas = [i/1800 for i in alphas]
plt.plot(lamdas, mses)
```



- Όσο μεγαλώνει το λ μεγαλώνει και το mse.
- Για λ από 0 έως 0,1, οι αυξήσεις στο mse είναι μικρές.

- Weights με βάση το Ridge **όταν λ=0.1**

Ridge Regression

```
]: # Import Ridge
from sklearn.linear_model import Ridge
# The alpha used by Python's ridge should be the lambda times the number of observ
# Here we produce results for alpha=180 which corresponds to lambda=0.1
ridge = Ridge(alpha=180)
ridge.fit(X_train, y_train)
# DataFrame with corresponding feature and its respective coefficients
coeffs = pd.DataFrame(
    [
        ['intercept'] + list(X_train.columns),
        list(ridge.intercept_) + list(ridge.coef_[0])
    ]
).transpose().set_index(0)


ridgereg_coefficient = pd.DataFrame()
ridgereg_coefficient["Features"] = X_train.columns
ridgereg_coefficient['Coef Estimate'] = pd.Series(ridge.coef_[0])
ridgereg_coefficient
```

Features	Weights	Features	Weights
LotArea	0.071	BrkSide	0.014
OverallQual	0.195	ClearCr	-0.004
OverallCond	0.076	CollgCr	-0.004
YearBuilt	0.104	Crawfor	0.033
YearRemodAdd	0.046	Edwards	-0.004
BsmtFinSF1	0.103	Gilbert	-0.010
BsmtUnfSF	-0.016	IDOTRR	-0.008
TotalBsmtSF	0.111	MeadowV	-0.017
1stFlrSF	0.122	Mitchel	-0.027
2ndFlrSF	0.079	Names	-0.034
GrLivArea	0.159	NoRidge	0.056
FullBath	0.007	NPkVill	-0.023
HalfBath	0.031	NriddgHt	0.119
BedroomAbvGr	-0.060	NWAmes	-0.047
TotRmsAbvGrd	0.081	OLDTown	-0.032
Fireplaces	0.042	SWISU	-0.009
GarageCars	0.047	Sawyer	-0.019
GarageArea	0.060	SawyerW	-0.024
WoodDeckSF	0.025	Somerst	0.027
OpenPorchSF	0.037	StoneBr	0.064
EnclosedPorch	0.003	Timber	0.000
Blmngtn	-0.016	Veenker	0.006
Blueste	-0.012	Bsmt Qual	0.037
BrDale	-0.021		

Διαλέγοντας το λ με βάση k - fold cross-validation (k=10)

Selecting alpha based on k - fold cross-validation

```
##Instead of arbitrarily choosing alpha  =4
## , it would be better to use cross-validation to choose the tuning parameter alpha. We can do this us
##Selecting Lambda
##Fit Ridge regression through cross validation

from sklearn.linear_model import RidgeCV
regr_cv= RidgeCV(alphas=alphas, fit_intercept=True, cv=10)
model_cv2=regr_cv.fit(X_train,y_train)
model_cv2.alpha_
```

```
135.0
```

```
# MSE at the validation dataset
ridge=Ridge(alpha=model_cv2.alpha_)
ridge.fit(X_train,y_train)
pred=ridge.predict(X_val)
print(mse(y_val,pred))
```

```
0.11825709631198018
```

```
# MSE at the test dataset
ridge=Ridge(alpha=model_cv2.alpha_)
ridge.fit(X_train,y_train)
pred=ridge.predict(X_test)
print(mse(y_test,pred))
```

```
0.12017808465302704
```

```
# Ridge
# import model
from sklearn.linear_model import Ridge
# Here we produce results for alpha=135 which corresponds to lambda=135/1800=0.075
ridge=Ridge(alpha=model_cv2.alpha_)
ridge.fit(X_train, y_train)

# DataFrame with corresponding feature and its respective coefficients
coeffs = pd.DataFrame(
    [
        ['intercept'] + list(X_train.columns),
        list(ridge.intercept_) + list(ridge.coef_[0])
    ]
).transpose().set_index(0)
ridgereg_coefficient = pd.DataFrame()
ridgereg_coefficient["Features"] = X_train.columns
ridgereg_coefficient['Coef Estimate'] = pd.Series(ridge.coef_[0])
ridgereg_coefficient
```

Features	Weights	Features	Weights
LotArea	0.073	BrkSide	0.015
OverallQual	0.199	ClearCr	-0.004
OverallCond	0.079	CollgCr	-0.004
YearBuilt	0.112	Crawfor	0.034
YearRemodAdd	0.042	Edwards	-0.003
BsmtFinSF1	0.103	Gilbert	-0.009
BsmtUnfSF	-0.018	IDOTRR	-0.007
TotalBsmtSF	0.114	MeadowV	-0.017
1stFlrSF	0.125	Mitchel	-0.027
2ndFlrSF	0.084	Names	-0.035
GrLivArea	0.164	NoRidge	0.056
FullBath	0.002	NPkVill	-0.023
HalfBath	0.029	NriddgHt	0.121
BedroomAbvGr	-0.064	NWAmes	-0.048
TotRmsAbvGrd	0.081	OLDTown	-0.032
Fireplaces	0.039	SWISU	-0.009
GarageCars	0.045	Sawyer	-0.019
GarageArea	0.059	SawyerW	-0.025
WoodDeckSF	0.024	Somerst	0.027
OpenPorchSF	0.037	StoneBr	0.064
EnclosedPorch	0.004	Timber	-0.001
Blmngtn	-0.016	Veenker	0.006
Blueste	-0.012	Bsmt Qual	0.033
BrDale	-0.022		

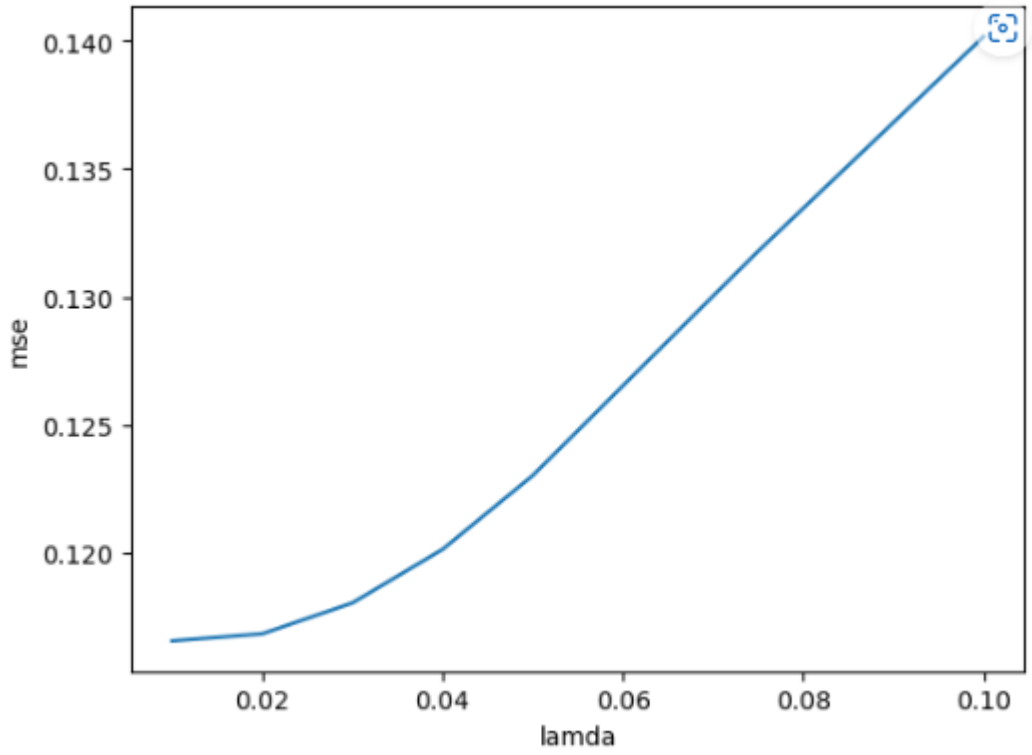
Αποτελέσματα - Lasso Regression

- Ανοίξτε το αρχείο python *6.3 lasso_regression.ipynb*
- Για το validation set

```
# We now consider different lambda values. The alphas are half the lambdas
alphas=[0.01/2, 0.02/2, 0.03/2, 0.04/2, 0.05/2, 0.075/2, 0.1/2]
mses=[]
for alpha in alphas:
    lasso=Lasso(alpha=alpha)
    lasso.fit(X_train,y_train)
    pred=lasso.predict(X_val)
    mses.append(mse(y_val,pred))
    print(mse(y_val, pred))
```

```
0.11654751909608799
0.11682687945311097
0.1180334835313203
0.12012836764959005
0.12301536903084047
0.13178576395045638
0.1401719458448378
```

```
plt.xlabel('lamda')
plt.ylabel('mse')
lamdas = [i *2 for i in alphas]
plt.plot(lamdas, mses)
```



- Non-zero weights με βάση το Lasso **όταν λ=0.1** (15 χαρακτηριστικά είναι τα πιο σημαντικά) - Variable selection

```
# Import Lasso
from sklearn.linear_model import Lasso
# Here we produce results for alpha=0.05 which corresponds to lambda=0.1
lasso = Lasso(alpha=0.05)
lasso.fit(X_train, y_train)
Lasso(alpha=0.05)
# DataFrame with corresponding feature and its respective coefficients
coeffs = pd.DataFrame(
    [
        ['intercept'] + list(X_train.columns),
        list(lasso.intercept_) + list(lasso.coef_)
    ]
).transpose().set_index(0)


lassoreg_coefficient = pd.DataFrame()
lassoreg_coefficient["Features"] = X_train.columns
lassoreg_coefficient['Coef Estimate'] = pd.Series(lasso.coef_)
lassoreg_coefficient
```

Feature	**Weight**
Lot Area (square feet)	0.04
Overall quality (Scale from 1 to 10)	0.30
Year built	0.05
Year remodeled	0.06
Finished basement (square feet)	0.12
Total basement (square feet)	0.10
First floor (square feet)	0.03
Living area (square feet)	0.30
Number of fireplaces	0.02
Parking spaces in garage	0.03
Garage area (square feet)	0.07
Neighborhoods (3 out of 25 non-zero)	0.01, 0.02, and 0.08
Basement quality	0.02

Διαλέγοντας το λ με βάση k - fold cross-validation (k=10)

```
##We can also select alpha based on the cross-validated ridge regression function, LassoCV().
##Selecting lambda
##Fit Lasso regression through cross validation

from sklearn.linear_model import LassoCV
regr_cv= LassoCV(alphas = None, cv = 10, max_iter = 100000)
model_cv2=regr_cv.fit(X_train,y_train)
model_cv2.alpha_
```

```
C:\Users\thano\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:1571: DataConversionWarning: A column-
ctor y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
0.005637136472209141
```

```
# MSE at the validation dataset
lasso=Lasso(alpha=model_cv2.alpha_)
lasso.fit(X_train,y_train)
pred=lasso.predict(X_val)
print(mse(y_val,pred))
```

```
0.11649288992474192
```

```
# MSE at the test dataset
lasso=Lasso(alpha=model_cv2.alpha_)
lasso.fit(X_train,y_train)
pred=lasso.predict(X_test)
print(mse(y_test,pred))
```

```
0.11886996279710277
```

```
# Here we produce results for alpha=0.006 which corresponds to lambda=0.012
lasso = Lasso(alpha=model_cv2.alpha_)
lasso.fit(X_train, y_train)

# DataFrame with corresponding feature and its respective coefficients
coeffs = pd.DataFrame(
    [
        ['intercept'] + list(X_train.columns),
        list(lasso.intercept_) + list(lasso.coef_)
    ]
).transpose().set_index(0)


lassoreg_coefficient = pd.DataFrame()
lassoreg_coefficient["Features"] = X_train.columns
lassoreg_coefficient['Coef Estimate'] = pd.Series(lasso.coef_)
lassoreg_coefficient
```

Features	Weights	Features	Weights
LotArea	0.077	Blueste	-0.005
OverallQual	0.228	BrDale	-0.014
OverallCond	0.082	BrkSide	0.023
YearBuilt	0.137	Crawfor	0.040
YearRemodAdd	0.032	Edwards	0.005
BsmtFinSF1	0.122	MeadowV	-0.004
TotalBsmtSF	0.107	Mitchel	-0.014
1stFlrSF	0.038	Names	-0.012
GrLivArea	0.302	NoRidge	0.055
HalfBath	0.018	NPkVill	-0.014
BedroomAbvGr	-0.069	NriddgHt	0.128
TotRmsAbvGrd	0.061	NWAmes	-0.035
Fireplaces	0.029	OLDTown	-0.011
GarageCars	0.034	SawyerW	-0.013
GarageArea	0.058	Somerst	0.029
WoodDeckSF	0.019	StoneBr	0.063
OpenPorchSF	0.031	Veenker	0.004
Blmngtn	-0.009	Bsmt Qual	0.015

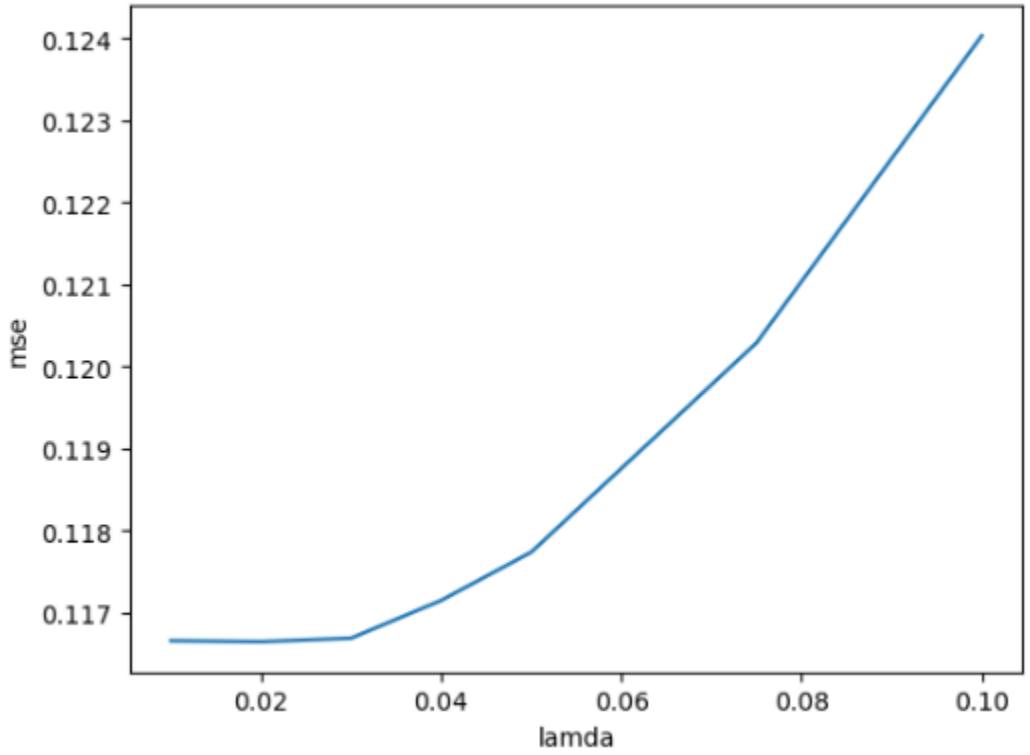
Αποτελέσματα - Elastic Net Regression

- Ανοίξτε το αρχείο python *6.4 elasticnet_regression.ipynb*
- Για το validation set

```
#ENET with different levels of alpha and its mse
# We now consider different lambda values. The alphas are half the lambdas
alphas=[0.01/2, 0.02/2, 0.03/2, 0.04/2, 0.05/2, 0.075/2, 0.1/2]
mses=[]
for alpha in alphas:
    e_net =ElasticNet(alpha=alpha)
    e_net.fit(X_train,y_train)
    pred=e_net.predict(X_val)
    mses.append(mse(y_val,pred))
    print(mse(y_val, pred))
```

```
0.11666265584936347
0.11665113418337304
0.1166935432601165
0.1171533394764451
0.11774118314106564
0.12028928749167629
0.12402805248512704
```

```
plt.xlabel('lamda')
plt.ylabel('mse')
lamdas = [i *2 for i in alphas]
plt.plot(lamdas, mses)
```



- Non-zero weights με βάση το Enet **όταν λ=0.1** (23 χαρακτηριστικά είναι τα πιο σημαντικά) - Variable selection

```
]: # Elastic Net
   # import model
   from sklearn.linear_model import ElasticNet

   # Here we produce results for alpha=0.05 which corresponds to lambda=0.1
   e_net = ElasticNet(alpha=0.05)
   e_net.fit(X_train, y_train)

   # DataFrame with corresponding feature and its respective coefficients
   coeffs = pd.DataFrame(
       [
           ['intercept'] + list(X_train.columns),
           list(e_net.intercept_) + list(e_net.coef_)
       ]
   ).transpose().set_index(0)


   enetreg_coefficient = pd.DataFrame()
   enetreg_coefficient["Features"] = X_train.columns
   enetreg_coefficient['Coef Estimate'] = pd.Series(e_net.coef_)
   enetreg_coefficient
```

Features	**Weights**
LotArea	0.06
OverallQual	0.26
OverallCond	0.03
YearBuilt	0.08
YearRemodAdd	0.06
BsmtFinSF1	0.12
TotalBsmtSF	0.09
1stFlrSF	0.05
GrLivArea	0.29
BedroomAbvGr	-0.01
TotRmsAbvGrd	0.01
Fireplaces	0.03
GarageCars	0.04
GarageArea	0.06
WoodDeckSF	0.01
OpenPorchSF	0.02
Crawfor	0.02
NoRidge	0.04
NriddgHt	0.11
NWAmes	-0.01
Somerst	0.01
StoneBr	0.05
Bsmt Qual	0.03

Διαλέγοντας το λ με βάση k - fold cross-validation

Selecting alpha based on cross-validation

```
##We can also select the alpha using the cross-validated ridge regression function, ElasticNetCV().
##Fit Enet regression through cross validation

from sklearn.linear_model import ElasticNetCV
regr_cv= ElasticNetCV(alphas = None, cv = 10, max_iter = 100000)
#regr_cv=LassoCV(alphas=range(1,50))
model_cv2=regr_cv.fit(X_train,y_train)
model_cv2.alpha_
```

```
C:\Users\thano\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:1571: DataConversionWarning: A
ctor y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel
  y = column_or_1d(y, warn=True)
```

```
0.011274272944418283
```

```
# MSE at the validation dataset
e_net=ElasticNet(alpha=model_cv2.alpha_)
e_net.fit(X_train,y_train)
pred=e_net.predict(X_val)
print(mse(y_val,pred))
```

```
0.11661058532779355
```

```
# MSE at the test dataset
e_net=ElasticNet(alpha=model_cv2.alpha_)
e_net.fit(X_train,y_train)
pred=e_net.predict(X_test)
print(mse(y_test,pred))
```

```
0.11919797316720483
```

```
# Elastic Net
# import model
from sklearn.linear_model import ElasticNet

# Here we produce results for alpha=0.011 which corresponds to lambda=0.022
e_net = ElasticNet(alpha=model_cv2.alpha_)
e_net.fit(X_train, y_train)

# DataFrame with corresponding feature and its respective coefficients
coeffs = pd.DataFrame(
    [
        ['intercept'] + list(X_train.columns),
        list(e_net.intercept_) + list(e_net.coef_)
    ]
).transpose().set_index(0)


enetreg_coefficient = pd.DataFrame()
enetreg_coefficient["Features"] = X_train.columns
enetreg_coefficient['Coef Estimate'] = pd.Series(e_net.coef_)
enetreg_coefficient
```

Features	Weights	Features	Weights
LotArea	0.077	Blueste	-0.005
OverallQual	0.227	BrDale	-0.014
OverallCond	0.080	BrkSide	0.023
YearBuilt	0.133	Crawfor	0.040
YearRemodAdd	0.034	Edwards	0.004
BsmtFinSF1	0.121	MeadowV	-0.004
TotalBsmtSF	0.104	Mitchel	-0.014
1stFlrSF	0.043	Names	-0.013
GrLivArea	0.293	NoRidge	0.055
HalfBath	0.020	NPkVill	-0.014
BedroomAbvGr	-0.066	NriddgHt	0.127
TotRmsAbvGrd	0.064	NWAmes	-0.035
Fireplaces	0.030	OLDTown	-0.012
GarageCars	0.035	SawyerW	-0.013
GarageArea	0.058	Somerst	0.029
WoodDeckSF	0.019	StoneBr	0.063
OpenPorchSF	0.031	Veenker	0.004
Blmngtn	-0.009	Bsmt Qual	0.018

Σύνοψη των αποτελεσμάτων

- **Χωρίς regularization υπάρχει overfitting.** Ορισμένα weights είναι αρνητικά όταν θα περιμέναμε να είναι θετικά. Δείτε το παράδειγμα της πολυμεταβλητής γραμμικής παλινδρόμησης.

- **Με βάση το k-fold cross-validation**

- Το Mean squared error για το test set για το Ridge με βάση το 10-fold είναι 12%, άρα εξηγείται το 88% της διακύμανσης.
- Το Mean squared error για το test set για το Lasso και το Enet με βάση το 10-fold είναι 11,9 %, άρα εξηγείται το 88,1% της διακύμανσης.
- Πολύ μικρές διαφορές μεταξύ Ridge, Lasso , ENet !

- Υπάρχουν και άλλα κριτήρια που μπορείτε να λάβετε υπόψιν για την επιλογή του λ.