Regression Models
Linear, Lasso and Ridge

Rao Vemuri

Problem Statement: Sankranti Sales

- Sankranti is coming
- Need to estimate / predict which product will sell well
- A bad decision can leave your customers to look for offers in competitor stores.
- The challenge does not end there – you need to estimate the sales of products across a range of different categories for stores in varied locations

Step 1: Make a List

- List all those factors you can think, on which the sales of a store will be dependent on.
- For each factor create a hypothesis about why and how that factor would influence the sales of various products.
  - Location of your shop
  - Size of the shop
  - Availability of the products
  - Offers on the product
  - Advertising done by a product
  - Placement in the store shelves, etc.

Big Market Sales

- Someone collected such data and is available for our use at
- https://datahack.analyticsvidhya.com/contest/practice-problem-big-mart-sales-iii/
-

Item_Weight	Item_Fat_Content	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales	Outlet_Size_Medium	Outlet_Size_Small	Outlet_Location_Type_Tier 2	O
9.3	0	-4.132214606	249.8092	14	3735.138	1	0	0	
5.92	1	-3.948779525	48.2692	4	443.4228	1	0	0	
17.5	0	-4.088755709	141.618	14	2097.27	1	0	0	
19.2	1	-2.716102099	182.095	15	732.38	0	1	0	
8.93	0	-2.716102099	53.8614	26	994.7052	0	0	0	
10.395	1	-2.716102099	51.4008	4	556.6088	1	0	0	
13.65	1	-4.362923154	57.6588	26	343.5528	0	0	0	
12.85764518	0	-2.059875358	107.7622	28	4022.7636	1	0	0	

Description of the Dataset

- In the dataset, we can see
  - characteristics of the sold item
  - (fat content, visibility, type, price)
  - characteristics of the outlet
  - (year of establishment, size, location, type)
  - the number of the items sold for that particular item.
- And many more
- Let’s see if we can predict sales using these features.

Model 1: Mean Sales

- Future sales = average of the past sales
- Good start! But, is it a good model?
- There are various ways in which we can evaluate how good our model is.
- The most common way is Mean Squared Error.
- Let us understand how to measure it.

- The simplest way of calculating error will be, to calculate the difference in the predicted and actual values.
- However, if we simply add them, they might cancel out, so we square these errors before adding.
- We also divide them by the number of data points to calculate a mean error since it should not be dependent on number of data points.

$$\frac{(e_1^2 + e_2^2 + \ldots + e_n^2)}{n}$$

The MSE

- Calculate the mean of all the data points
- Then calculate the error at each data point: mean - data
- Square it
- Add all the squared errors
- Divide by the number of data points
- The MSE = 29,11,799
- Wow!! That is huge

Model 2: Average Sales by Location

- Repeat the procedure by breaking down the data into locations:
- Mumbai
- Chennai
- Delhi
- Vizag
- By doing this we get
- MSE = 28, 75, 386
- A little better. Not a great Deal

Model 3: Liner Regression

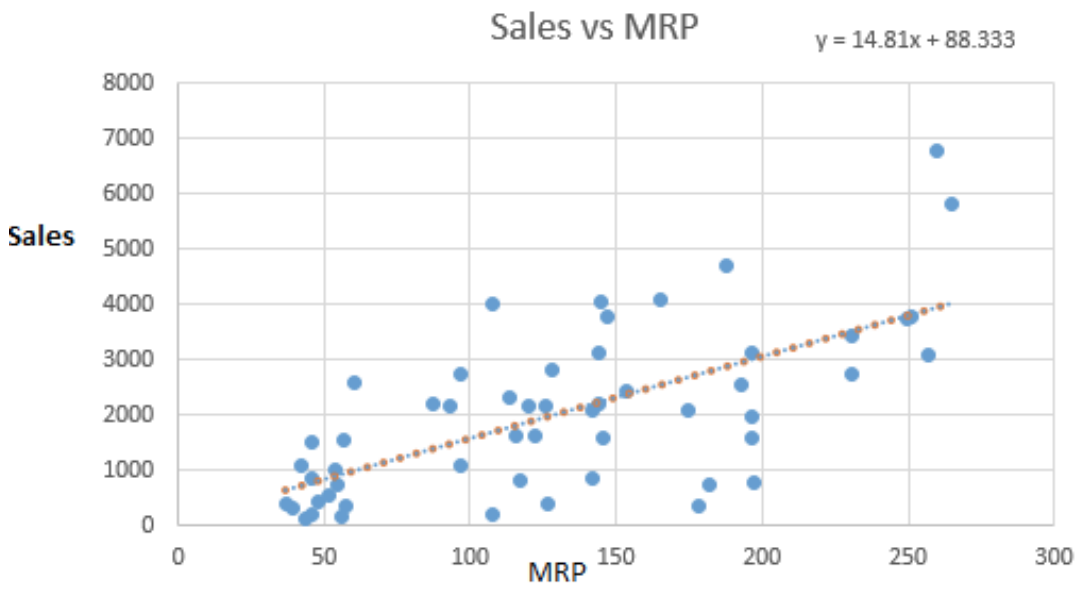
- Let us try a linear model of the type

$$Y = \theta_1 X_1 + \theta_2 X_2 + \ldots \theta_n X_n$$

- Y = sales
- X's are features
- Thetas: importance of the features (weights)
- Let us try a simple case (one feature, viz MRP)

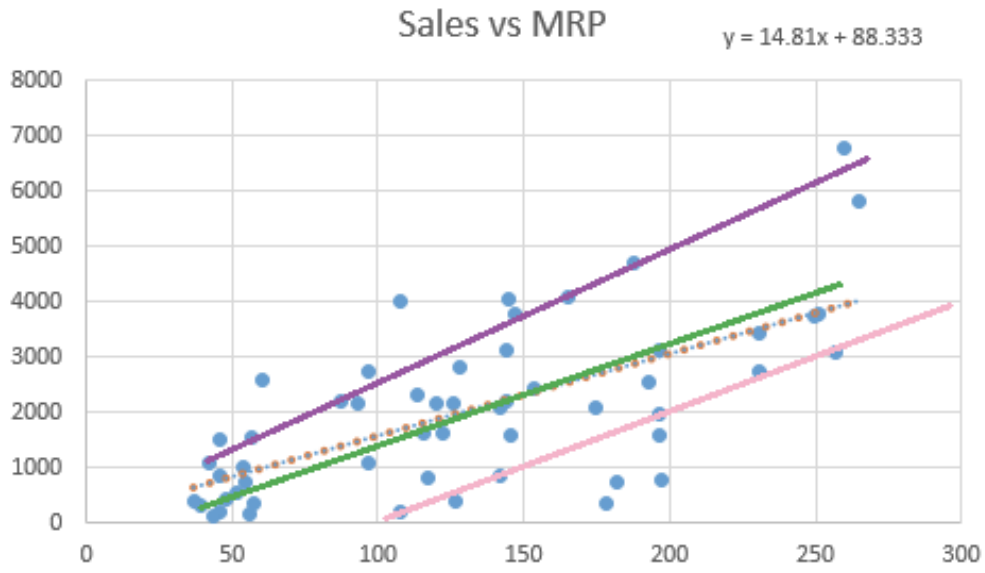
$$Y = \theta_1 * X + \theta_0$$

Scatter Plot



- Surprisingly, we can see that sales of a product increases with increase in its MRP.
- Therefore the dotted red line represents our regression line or the line of best fit.
- But question is: How do you find this line?

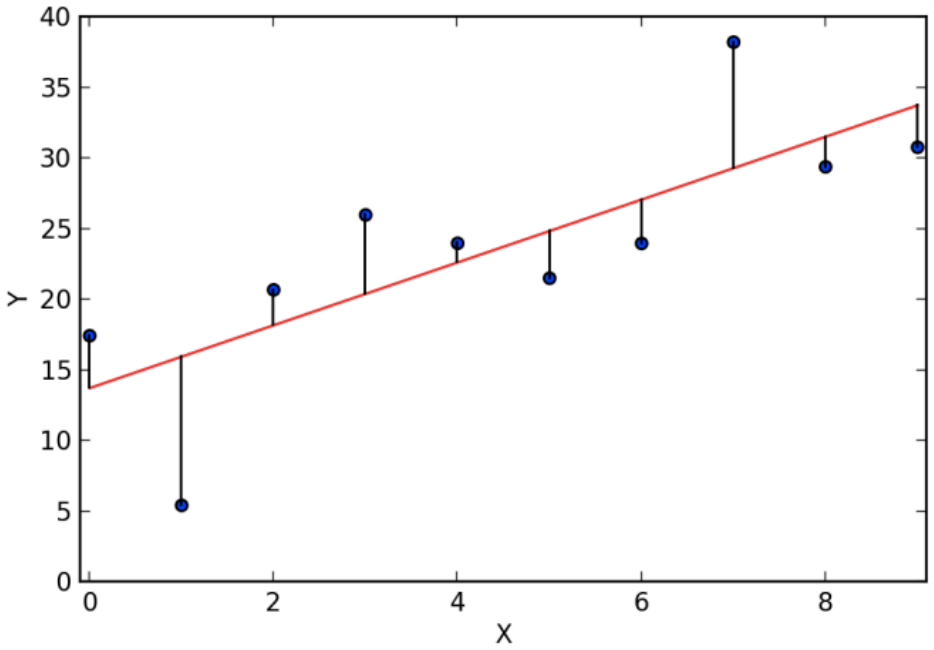
The Line of Best Fit



Purpose of Best Fit

- The main purpose of the best fit line is that values predicted by our model should be close to the observed values, because there is no point in predicting values which are far away from the real values.
- In other words, we tend to minimize the difference between the values predicted by us and the observed values, termed as error.
- Graphical representation of error is as shown below. These errors are also called as residuals. The residuals are indicated by the vertical lines showing the difference between the predicted and actual value.

Errors or Residuals



How to Define Error

- **Sum of residuals ($\sum(Y - h(X))$)** – it might result in cancelling out of positive and negative errors.
- **Sum of the absolute value of residuals ($\sum|Y-h(X)|$)** – absolute value would prevent cancellation of errors. But you cannot find the derivative everywhere.
- **Sum of square of residuals ($\sum (Y-h(X))^2$)** – it's the method mostly used in practice since here we penalize higher error value much more as compared to a smaller one, so that there is a significant difference between making big errors and small errors, which makes it easy to differentiate and select the best fit line.There are other reasons too.

SS Residuals = J

$$SS_{residuals} = \sum_{i=1}^{m} (h(x) - y)^2$$

where, h(x) is the value predicted by us, $h(x) = \Theta_1 * x + \Theta_0$,
y is the actual values and
m is the number of rows in the training set.

Cost

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right)^2$$

Gradient Descent

- Model 3: Linear Regression with Gradient Descent

Import libraries

```
# importing basic libraries
import numpy as np
import pandas as pd
from pandas import Series, DataFrame
from sklearn.model_selection import train_test_split
import test and train file
train = pd.read_csv(‘train.csv')
test = pd.read_csv('test.csv')
```

Import Linear Regression from Sklearn

```
# importing linear regression from sklearn
from sklearn.linear_model import LinearRegression
lreg = LinearRegression()
#splitting into training and cv for cross validation
X = train.loc[:,['Outlet_Establishment_Year','Item_MRP']]
x_train, x_cv, y_train, y_cv = train_test_split(X,train.Item_Outlet_Sales)
```

Training the Model

```
#training the model
lreg.fit(x_train, y_train)
#predicting on cv
pred = lreg.predict(x_cv)
#calculating mse
mse = np.mean((pred - y_cv)**2)
# we got mse = 19,10,586.53
```

Coefficients of Regression Model

#Let us take a look at the coefficients of this linear regression model.

calculating coefficients

```
coeff = DataFrame(x_train.columns)
coeff['Coefficient Estimate'] = Series(lreg.coef_)
Coeff
```

	0	Coefficient Estimate
0	Outlet_Establishment_Year	-11.892070
1	Item_MRP	15.484779

we can see that MRP has a high coefficient, meaning items having higher prices have better sales.

Evaluating the Model: R-squared Statistic

- How accurate is the model? Do we have any evaluation metric, so that we can check this? Actually we have a quantity, known as R-Square.
- R-square is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression.
- The definition of R-square is fairly straight-forward; it is the percentage of the response variable variation that is explained by a linear model. Or:
- R-square = Explained variation / Total variation

Evaluating the Model- R^2

- **R-Square**: Determines how much of the total variation in y (dependent variable) is explained by the variation in x (independent variable). Mathematically, it can be written as:

$$R^2 = 1 - \frac{\sum [y_{actual} - y_{predicted}]^2}{\sum [y_{actual} - y_{mean}]^2}$$

- The value of R-square is always between 0 and 1, where 0 means that the model does not explain any variability in the target variable *y* and 1 meaning it explains full variability in *y*

Meaning of R^2

- Now let us check the R-square for the above model.

```
lreg.score(x_cv, y_cv)
```

0.3287

- In this case, R^2 is 32%, meaning, only 32% of variance in sales is explained by year of establishment and MRP.
- In other words, if you know the year of establishment and the MRP, you'll have 32% information to make an accurate prediction about its sales.

Limitations of R-square

- R-square does *not* indicate whether a regression model is adequate.
- You can have a low R-squared value for a good model, or a high R-squared value for a model that does not fit the data!
- The R-square in your regression output is a biased estimate based on your sample.
- An unbiased estimate is one that is just as likely to be too high as it is to be too low, and it is correct on average. If you collect a random sample correctly, the sample mean is an unbiased estimate of the population mean.
- A biased estimate is systematically too high or low, and so is the average.
- It's like a bathroom scale that always indicates you are heavier than you really are. No one wants that!

Model 4: LR with Three Features

#Let us introduce another feature ‘weight’. Now let’s build a regression model with these three features.

```
X = train.loc[:,['Outlet_Establishment_Year','Item_MRP','Item_Weight']]
```

#splitting into training and cv for cross validation

```
x_train, x_cv, y_train, y_cv = train_test_split(X,train.Item_Outlet_Sales)
```

training the model

```
lreg.fit(x_train,y_train)
```

- ValueError: Input contains NaN, infinity or a value too large for dtype(‘float64’).

Impute Missing Values

- It produces an error, because item weights column have some missing values. So let us impute them with the mean of other non-null entries.

```
train['Item_Weight'].fillna((train['Item_Weight'].mean()), inplace=True)
#training the model lreg.fit(x_train, y_train) again
## splitting into training and cv for cross validation
x_train, x_cv, y_train, y_cv = train_test_split(X,train.Item_Outlet_Sales)
## training the model
lreg.fit(x_train, y_train)
```

Predicting

```
#predicting on cv
pred = lreg.predict(x_cv)
calculating mse
mse = np.mean((pred - y_cv)**2)
mse
1853431.59
```

Motivation for Adjusted R^2

calculating coefficients

coeff = DataFrame(x_train.columns)

coeff['Coefficient Estimate'] = Series(lreg.coef_)

	0	Coefficient Estimate
0	Outlet_Establishment_Year	-10.933946
1	Item_MRP	15.650872
2	Item_Weight	1.203274

#calculating r-square

lreg.score(x_cv,y_cv)

0.32942

- mse is further reduced. There is an increase in the value R-square. Does it mean that the addition of item weight is useful for our model?

Adjusted R^2

- The only drawback of R^2 is that if new predictors *x* are added to our model, R^2 only increases or remains constant but it never decreases. So it is hard to make judgment. Are we making it more accurate?
- That is why, we use “Adjusted R-Square”.
- The Adjusted R-Square is the modified form of R-Square that has been adjusted for the number of predictors in the model. It incorporates model’s degrees of freedom.
- The adjusted R-Square only increases if the new term improves the model accuracy.

Formula for Adjusted R^2

$$R^2 \text{ adjusted} = 1 - \frac{(1 - R^2)(N - 1)}{N - p - 1}$$

Where

R^2 = Sample R square
p = Number of predictors
N = total sample size

Using *all* Features

- Now let us build a model containing all the features.
- So far we have only used continuous features.
- But we have categorical features as well. We have to treat them differently
- There are different techniques to treat them
- Here we are using one-hot encoding (convert each class of a categorical variable as a feature).
- Also, we have imputed the missing values for “outlet size.”

Step 1: Imputing Missing Values

```
# imputing missing values
train['Item_Visibility'] =
train['Item_Visibility'].replace(0,np.mean(train['Item_Visibility
']))
train['Outlet_Establishment_Year'] = 2013 -
train['Outlet_Establishment_Year']
train['Outlet_Size'].fillna('Small',inplace=True)
```

Step 2: Categorical → Numerical

```
# creating dummy variables to convert categorical into numeric values
mylist = list(train1.select_dtypes(include=['object']).columns)
dummies = pd.get_dummies(train[mylist], prefix= mylist)
train.drop(mylist, axis=1, inplace = True)
X = pd.concat([train,dummies], axis =1 )
```

Step3: Building Model (Import libraries)

```
import numpy as np
import pandas as pd
from pandas import Series, DataFrame
import matplotlib.pyplot as plt
%matplotlib inline
train = pd.read_csv('training.csv')
test = pd.read_csv('testing.csv')
```

Import LR and Training/Test Split

```
# importing linear regression
from sklearn from sklearn.linear_model import LinearRegression
lreg = LinearRegression()
# for cross validation
from sklearn.model_selection import train_test_split
X = train.drop('Item_Outlet_Sales',1)
x_train, x_cv, y_train, y_cv = train_test_split(X,train.Item_Outlet_Sales,
test_size =0.3)
```

Training LR Model

```
# training a linear regression model on train
lreg.fit(x_train,y_train)
# predicting on cv
pred_cv = lreg.predict(x_cv)
# calculating mse
mse = np.mean((pred_cv - y_cv)**2)
mse
```

1348171.96

Evalauation Using R2

evaluation using r-square

lreg.score(x_cv,y_cv)

0.5483154146087005 9

- Clearly, we can see that there is a great improvement in both mse and R-square, which means that our model now is able to predict much closer values to the actual values.

Selecting the Right Feature Set

- It would be highly inefficient to use all the features since some of them might be imparting redundant information.
- We would need to select the right set of features which give us an accurate model as well as be able to explain the dependent variable well.
- There are multiple ways to select the right set of features for the model. For instance while predicting sales we know that marketing efforts should impact positively towards sales and is an important feature in our model.
- We should also take care that the variables we're selecting should not be correlated among themselves.

Automatic Feature Selection

- Instead of manually selecting the variables, we can automate this process by using forward or backward selection.
- Forward selection starts with most significant predictor in the model and adds variable for each step.
- Backward elimination starts with all predictors in the model and removes the least significant variable for each step.
- Selecting criteria can be set to any statistical measure like R-square, t-stat etc.
- We can use Deep Learning for feature selection

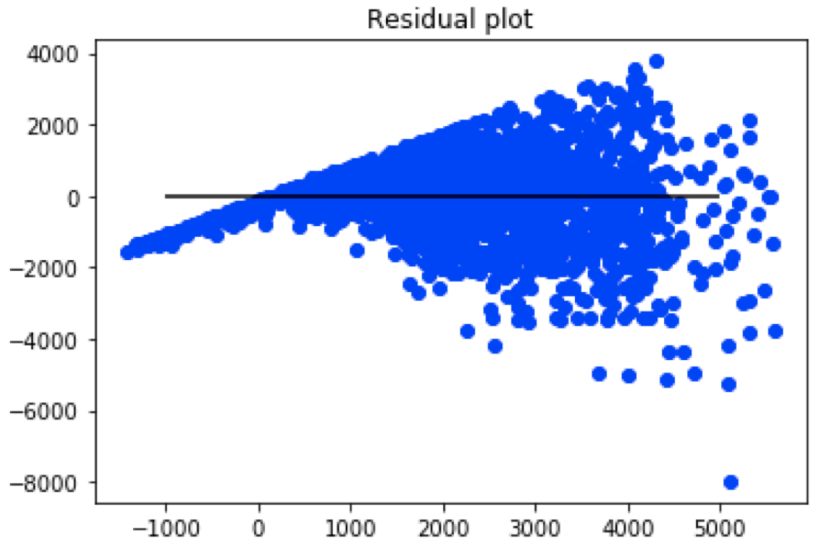
Check for Heteroskedasticity

- It is customary to check for heteroskedasticity of residuals once you build the linear regression model.
- The reason is, we want to check if the model thus built is unable to explain some pattern in the response variable Y, that eventually shows up in the residuals.

```
#residual plot
x_plot = plt.scatter(pred_cv, (pred_cv - y_cv), c='b')
plt.hlines(y=0, xmin= -1000, xmax=5000)
plt.title('Residual plot')
```

Residual Plot

```
<matplotlib.text.Text at 0x6e915433c8>
```



Residual vs Fitted Values Plot

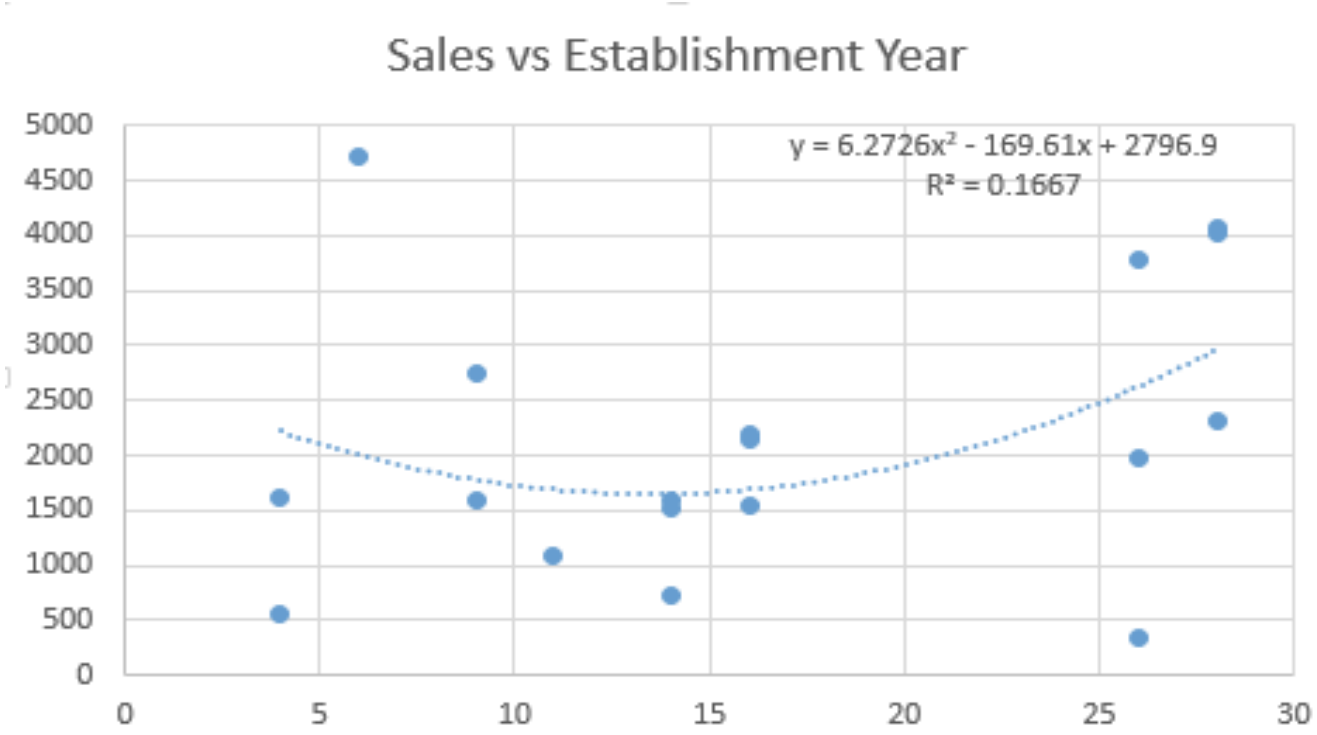
- The presence of non-constant variance in the error terms results in heteroskedasticity, which is indicated by the funnel shape.
- Generally, non-constant variance arises in presence of outliers or extreme leverage values. These values get too much weight, thereby disproportionately influencing the model's performance.
- This indicates signs of non linearity in the data which has not been captured by the model.
- To capture the non-linear effects, we have to use non-linear regression

Polynomial Regression

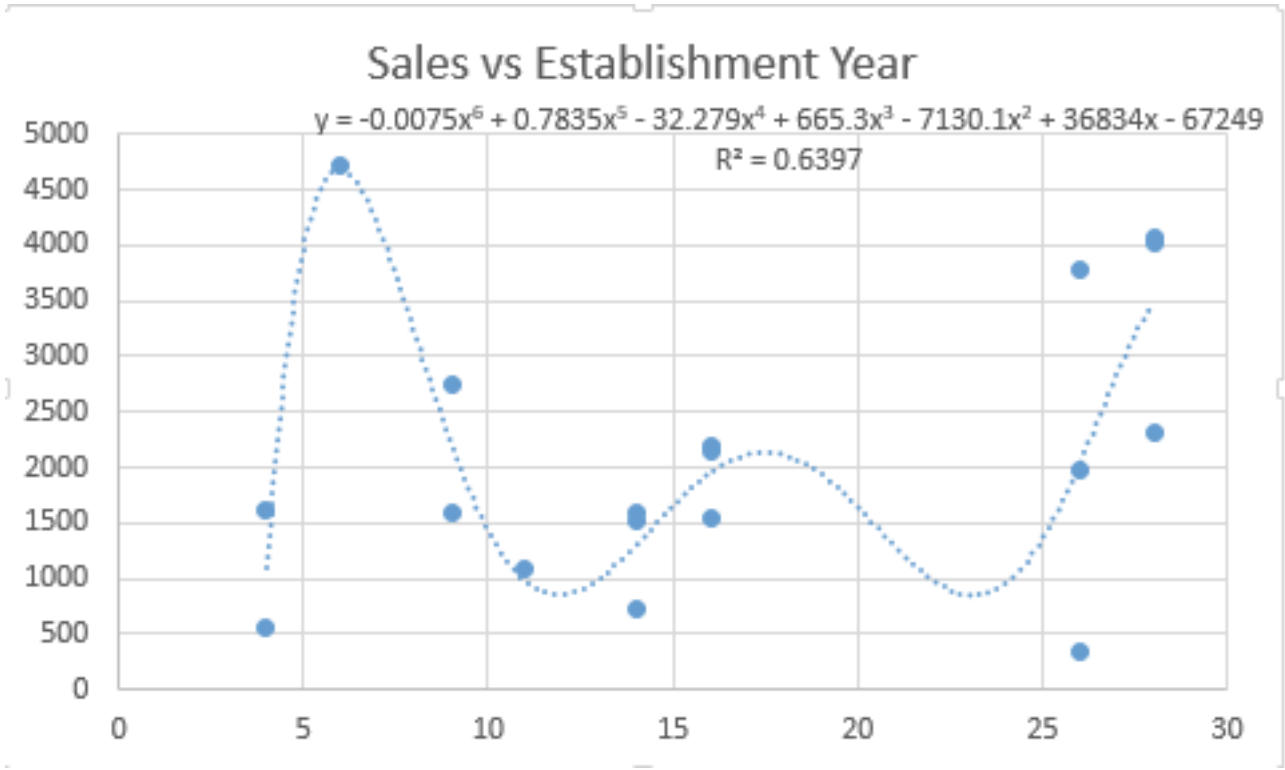
- Polynomial regression is another form of regression in which the maximum power of the independent variable is more than 1. In this regression technique, the best fit line is not a straight line instead it is in the form of a curve.
- Quadratic regression, or regression with second order polynomial, is given by the following equation:

$Y = \Theta_1 + \Theta_2 * x + \Theta_3 * x^2$

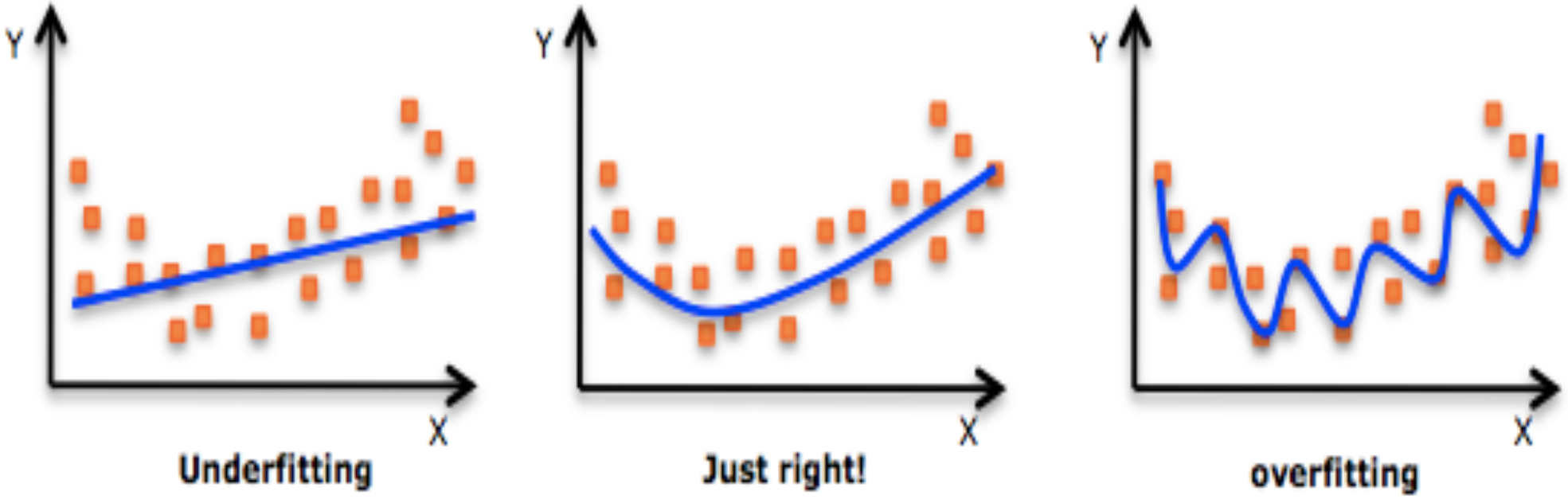
Quadratic Regression



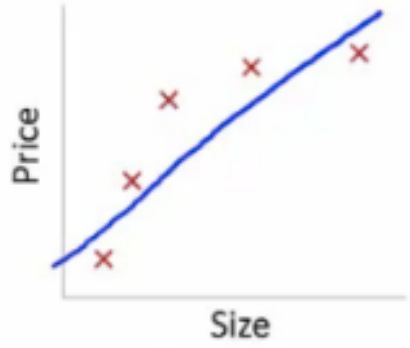
6th degree Polynomial Regression



Under-fitting Vs Over-fitting

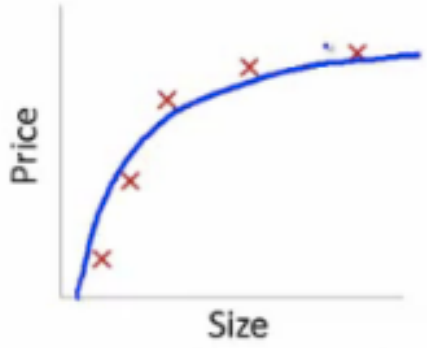


Bias-Variance



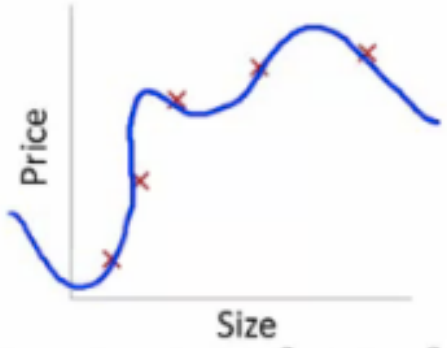
$\theta_0 + \theta_1 x$

High bias
(underfit)



$\theta_0 + \theta_1 x + \theta_2 x^2$

"Just right"



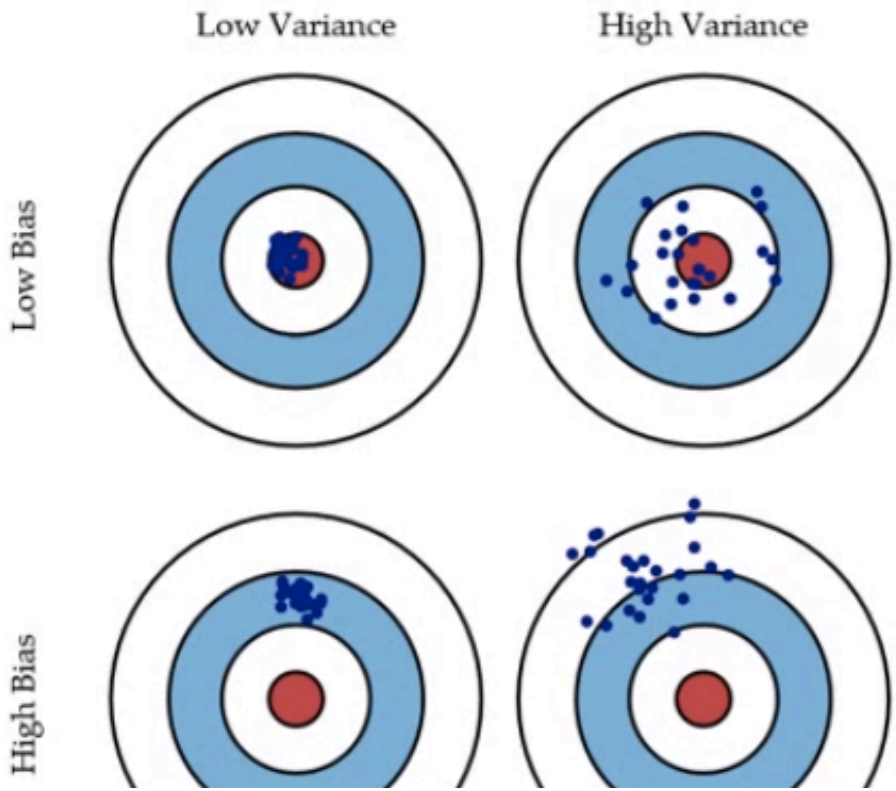
$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

High variance
(overfit)

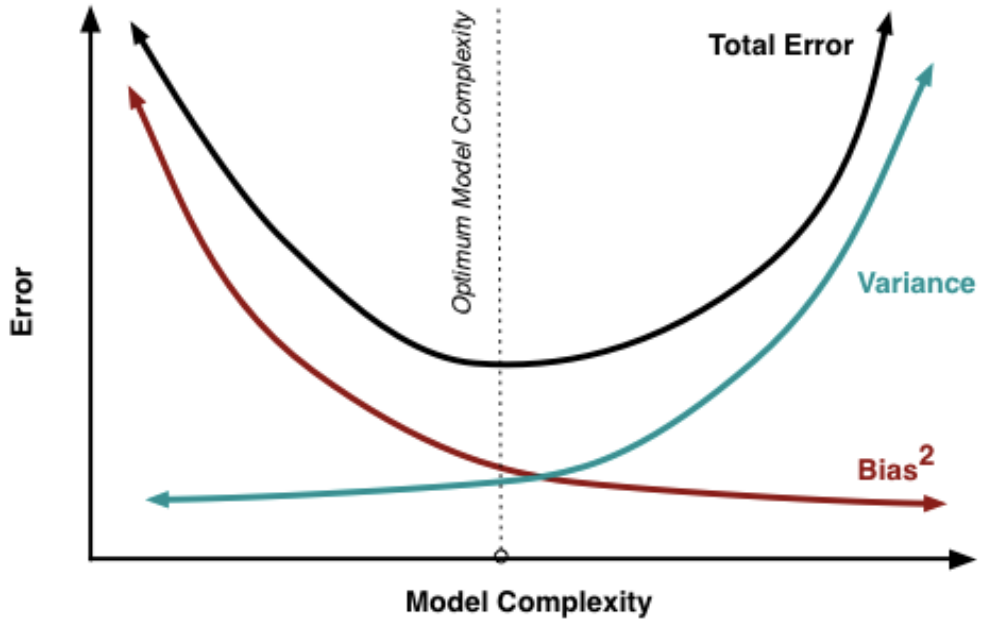
Over-Fitting/Under-Fitting

- Is it always better to use higher-order polynomials to fit the dataset?
- Sadly, no. Basically, we have created a model that fits our training data well but fails to estimate the real relationship among variables beyond the training set. Therefore our model performs poorly on the test data.
- This problem is called as **over-fitting**. We also say that the model has **high variance and low bias**.
- The model is **underfit** when we have **high bias and low variance**.

Bias-Variance Dilemma



Model Complexity Vs Error



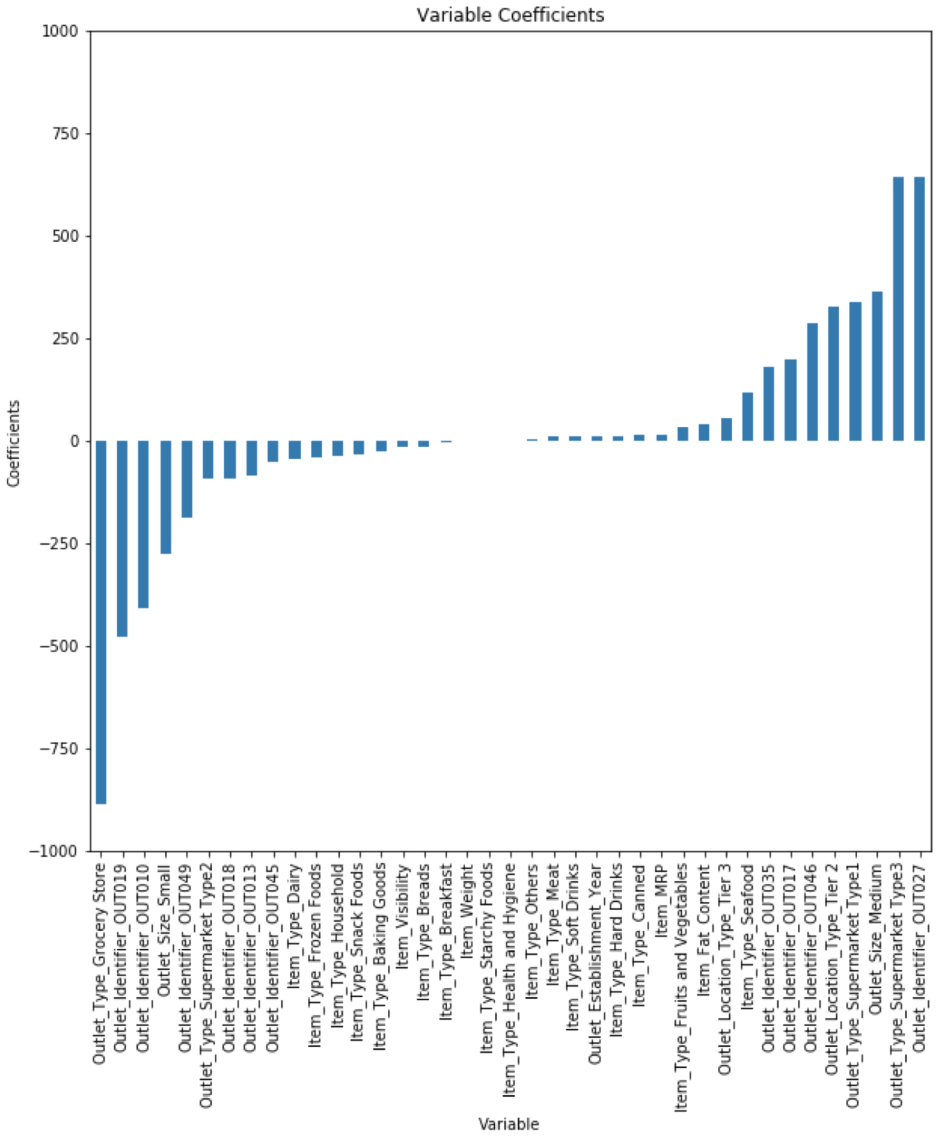
What is Regularization

- Regularization is an effort to avoid complex explanations even though they fit observed data better.
- Complex explanations do not generalize well.
- That is, they may explain past data well, but not future (unseen) data.
- Regularization discourages complex explanations even if they fit the data well and prefers simpler explanations even if they occasionally make a mistake.
- This means:
  - do not load your model with too many features
  - Do not over-fit with higher order nonlinearities

How to do Regularization

- Do not introduce nonlinear terms just to fit the data
- Reduce the number of features
- Keep the same number of features, but reduce the weights
- Let us look at the coefficients of features in our regression model.

```
predictors = x_train.columns
coef = Series(lreg.coef_,predictors).sort_values()
coef.plot(kind='bar', title='Modal Coefficients')
```


Variable Coefficients
Coefficients
1000
750
500
250
0
−250
−500
−750
−1000
Outlet_Type_Grocery Store
Outlet_Identifier_OUT019
Outlet_Identifier_OUT010
Outlet_Size_Small
Outlet_Identifier_OUT049
Outlet_Type_Supermarket Type2
Outlet_Identifier_OUT018
Outlet_Identifier_OUT013
Outlet_Identifier_OUT045
Item_Type_Dairy
Item_Type_Frozen Foods
Item_Type_Household
Item_Type_Snack Foods
Item_Type_Baking Goods
Item_Visibility
Item_Type_Breads
Item_Type_Breakfast
Item_Weight
Item_Type_Starchy Foods
Item_Type_Health and Hygiene
Item_Type_Others
Item_Type_Meat
Item_Type_Soft Drinks
Outlet_Establishment_Year
Item_Type_Hard Drinks
Item_Type_Canned
Item_MRP
Item_Type_Fruits and Vegetables
Item_Fat_Content
Outlet_Location_Type_Tier 3
Item_Type_Seafood
Outlet_Identifier_OUT035
Outlet_Identifier_OUT017
Outlet_Identifier_OUT046
Outlet_Location_Type_Tier 2
Outlet_Type_Supermarket Type1
Outlet_Size_Medium
Outlet_Type_Supermarket Type3
Outlet_Identifier_OUT027
Variable

Observation

- We can see that coefficients of Outlet_Identifier_OUT027 and Outlet_Type_Supermarket_Type3 (last 2) is much higher as compared to rest of the coefficients. Therefore the total sales of an item would be more driven by these two features.
- How can we reduce the magnitude of coefficients in our model?
- For this purpose, we have different types of regression techniques which uses regularization to overcome this problem.
  - Ridge Regression or L2 regularization
  - LASSO Regression or L1 regularization

Ridge Regression

- In linear regression, we estimate y by minimizing the sum of the squares of the errors.
- In ridge regression, we estimate y by minimizing the sum of the squares of the errors while satisfying the constraint that the squared sum of the weights is less than a prescribed quantity.
- RR chooses the chooses the weights in such a way that less influential features undergo more penalization.
- In some domains, the number of independent variables is many, as well as we are not sure which of the independent variables influences dependent variable. In this kind of scenario, ridge regression plays a better role than linear regression.

Mathematically

For two features:

$$Min \sum_{i=1}^{n} \left(y_i - (w_0 + w_1 x_{i1} + w_2 x_{i2})\right)^2$$

Subject to

$$\sum_{j=1}^{p} \left(w_j^2\right) \leq c$$

Using Lagrange multiplier $\alpha, we\ get$ (after ignoring αc in Min operation)

$$Min \sum_{i=1}^{n} \left(y_i - (w_0 + w_1 x_{i1} + w_2 x_{i2})\right)^2 + \alpha\left(\sum_{j=1}^{p}\left(w_j^2\right)\right)$$

Implementation of Ridge Regression

```
#from sklearn.linear_model import Ridge
## training the model
ridgeReg = Ridge(alpha=0.05, normalize=True)
ridgeReg.fit(x_train,y_train)
pred = ridgeReg.predict(x_cv)
#calculating mse
mse = np.mean((pred_cv - y_cv)**2)
mse
1348171.96
## calculating score
ridgeReg.score(x_cv,y_cv)
0.5691
```

Comments

- we can see that there is a slight improvement in our model because the value of the R-Square has been increased from 0.548...to 0.5691
- Note that value of alpha, which is a hyperparameter of Ridge Regression, (ie, it is not automatically learned by the model, instead we have to set it manually.
- How do we know the best value for alpha?
- By trial and error. You try several values of alpha and see when the value of R-square is a maximum. In this case, it occurs at alpha = 0.05
- This method is also called L2 regression

Lasso Regression

- Mathematics behind lasso regression is quiet similar to that of ridge
- Instead of adding squares of *w*, we will add absolute value of *w*

$$Min\ \sum_{i=1}^{n}\left(\mathrm{y_i} - (\mathrm{w_0} + \mathrm{w_1 x_{i1}} + \mathrm{w_2 x_{i2}})\right)^2 + \alpha\left(\sum_{j=1}^{p}\left(w_j^2\right)\right)$$

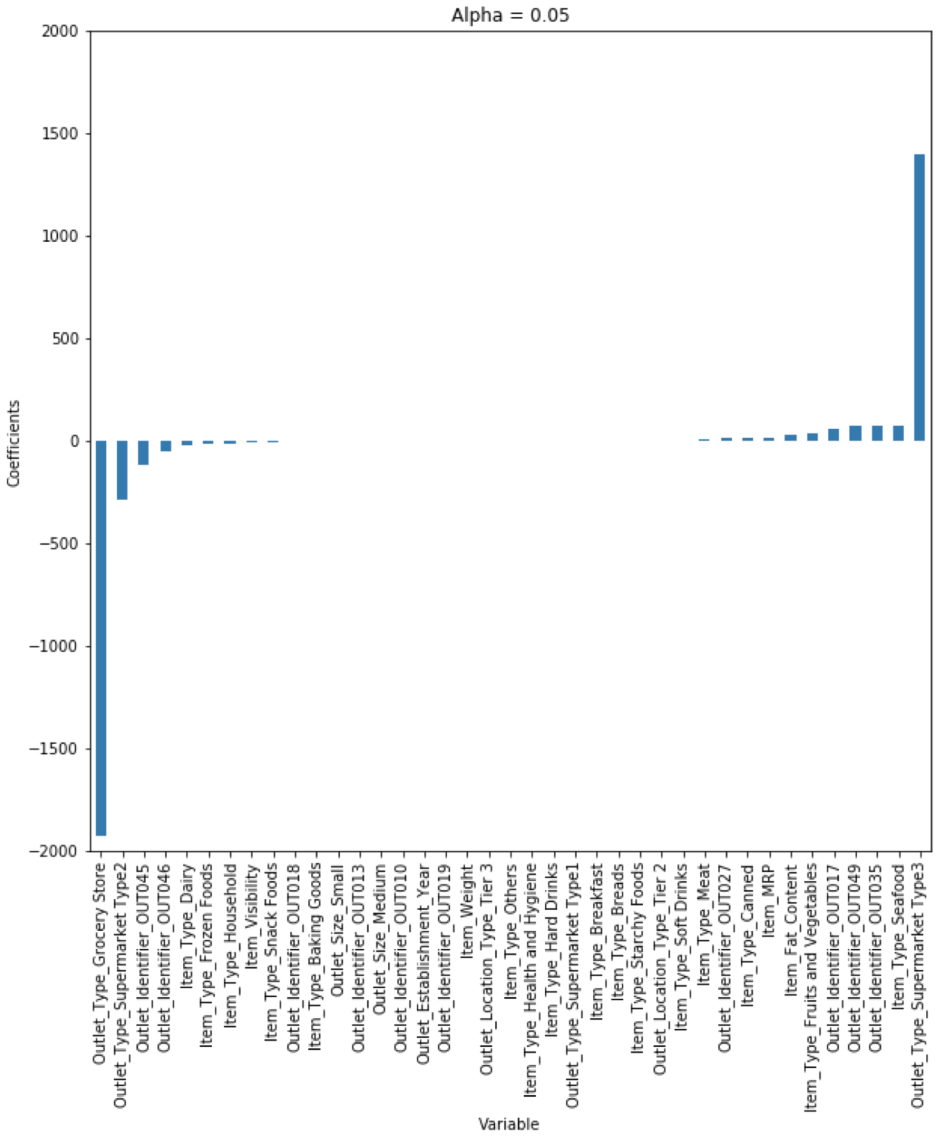
- *Also called L1 regression*
- LASSO (Least Absolute Shrinkage Selector Operator)

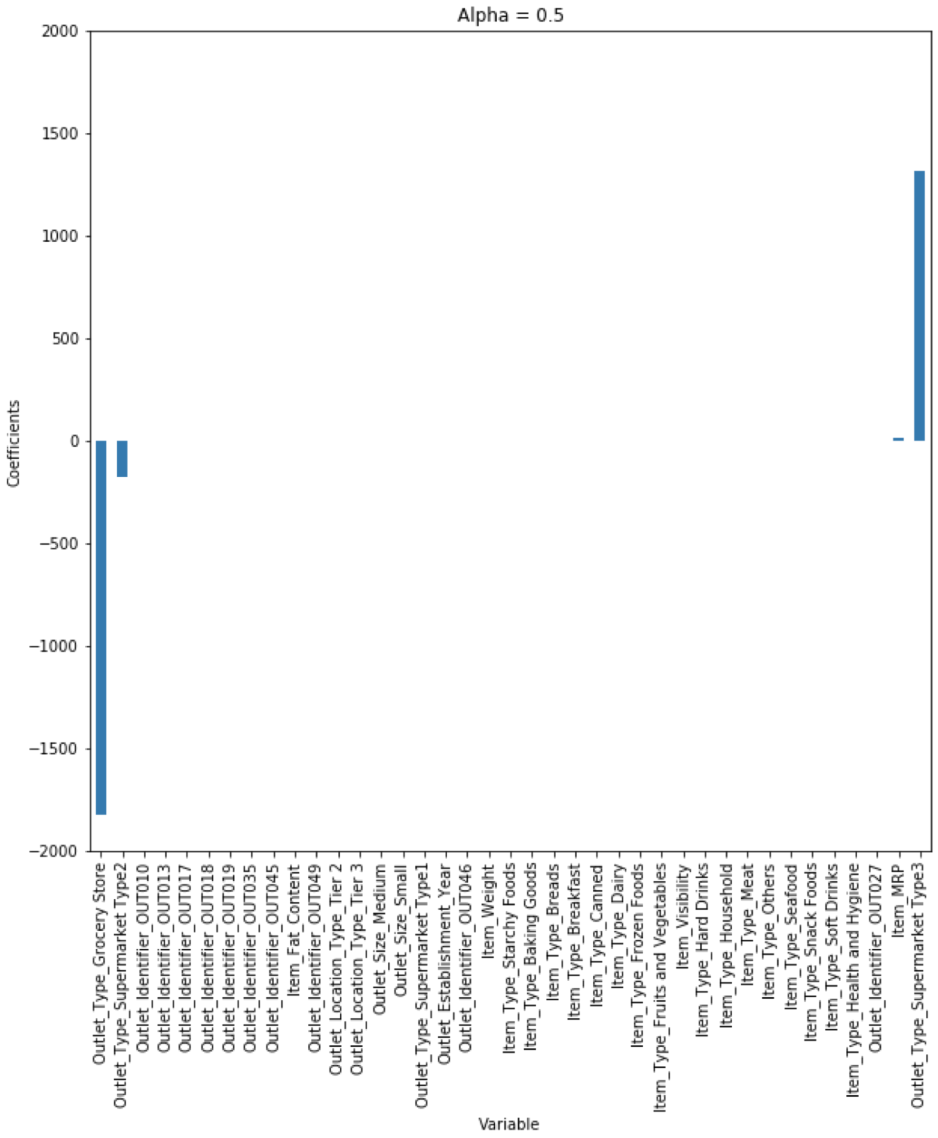
LASSO in Scikit

```
from sklearn.linear_model import Lasso
lassoReg = Lasso(alpha=0.3, normalize=True)
lassoReg.fit(x_train,y_train)
pred = lassoReg.predict(x_cv)
# calculating mse
mse = np.mean((pred_cv - y_cv)**2)
mse
1346205.82
lassoReg.score(x_cv,y_cv)
0.5720
```

Comments

- See that both the mse and the value of R-square have been increased.
- Therefore, lasso model is predicting better than both linear and ridge.
- Let us study the effect of changing alpha on the parameters by plotting (as we did for Ridge) for alpha = 0.05 and alpha = 0.5


Alpha = 0.05
2000
1500
1000
500
0
−500
−1000
−1500
−2000
Coefficients
Outlet_Type_Grocery Store
Outlet_Type_Supermarket Type2
Outlet_Identifier_OUT045
Outlet_Identifier_OUT046
Item_Type_Dairy
Item_Type_Frozen Foods
Item_Type_Household
Item_Visibility
Item_Type_Snack Foods
Outlet_Identifier_OUT018
Item_Type_Baking Goods
Outlet_Size_Small
Outlet_Identifier_OUT013
Outlet_Size_Medium
Outlet_Identifier_OUT010
Outlet_Establishment_Year
Outlet_Identifier_OUT019
Item_Weight
Outlet_Location_Type_Tier 3
Item_Type_Others
Item_Type_Health and Hygiene
Item_Type_Hard Drinks
Outlet_Type_Supermarket Type1
Item_Type_Breakfast
Item_Type_Breads
Item_Type_Starchy Foods
Outlet_Location_Type_Tier 2
Item_Type_Soft Drinks
Item_Type_Meat
Outlet_Identifier_OUT027
Item_Type_Canned
Item_MRP
Item_Fat_Content
Item_Type_Fruits and Vegetables
Outlet_Identifier_OUT017
Outlet_Identifier_OUT049
Outlet_Identifier_OUT035
Item_Type_Seafood
Outlet_Type_Supermarket Type3
Variable


Alpha = 0.5
Coefficients
2000
1500
1000
500
0
−500
−1000
−1500
−2000
Outlet_Type_Grocery Store
Outlet_Type_Supermarket Type2
Outlet_Identifier_OUT010
Outlet_Identifier_OUT013
Outlet_Identifier_OUT017
Outlet_Identifier_OUT018
Outlet_Identifier_OUT019
Outlet_Identifier_OUT035
Outlet_Identifier_OUT045
Item_Fat_Content
Outlet_Identifier_OUT049
Outlet_Location_Type_Tier 2
Outlet_Location_Type_Tier 3
Outlet_Size_Medium
Outlet_Size_Small
Outlet_Type_Supermarket Type1
Outlet_Establishment_Year
Outlet_Identifier_OUT046
Item_Weight
Item_Type_Starchy Foods
Item_Type_Baking Goods
Item_Type_Breads
Item_Type_Breakfast
Item_Type_Canned
Item_Type_Dairy
Item_Type_Frozen Foods
Item_Type_Fruits and Vegetables
Item_Visibility
Item_Type_Hard Drinks
Item_Type_Household
Item_Type_Meat
Item_Type_Others
Item_Type_Seafood
Item_Type_Snack Foods
Item_Type_Soft Drinks
Item_Type_Health and Hygiene
Outlet_Identifier_OUT027
Item_MRP
Outlet_Type_Supermarket Type3
Variable

Comments

- As we increased alpha from 0.05 to 0.5, the parameter (theta) values came down almost to zero
- This means, lasso selects only some feature as it reduces the coefficients of others to zero.
- This property is known as feature selection and which is absent in case of ridge.
- So lasso is generally used when we have more features, because it automatically does feature selection.

Ridge or Lasso?

- Suppose our problem has 10,000 features.
- Assume we know that some of the independent features are correlated with other independent features.
- Then which regression would you use, Ridge or Lasso?
- If we apply ridge regression, we retain all of the features but will shrink the coefficients. The model still remains complex as there are 10,000 features, thus may lead to poor model performance.
- When we have correlated variables, Lasso retains only one variable and sets other correlated variables to zero. That will possibly lead to some loss of information resulting in lower accuracy in our model.
- What to do? Use both

Elastic Net Regression

```
from sklearn.linear_model import ElasticNet
ENreg = ElasticNet(alpha=1, l1_ratio=0.5, normalize=False)
ENreg.fit(x_train,y_train)
pred_cv = ENreg.predict(x_cv)
#calculating mse
mse = np.mean((pred_cv - y_cv)**2)
mse 1773750.73
ENreg.score(x_cv,y_cv)
0.4504
```

Comments

- R-square is now much less than Ridge and Lasso.
- Why?
- Because we used both Ridge and Lasso
  - now have far fewer features
- The equation we used now is

$$\min\left(\left|\left|\mathrm{Y}-\mathrm{X}\theta\right|\right|_2^2+\lambda_1||\theta||_1+\lambda_2||\theta||_2^2\right)$$

- Why this works?
- What are the two parameters in the code?: alpha and L1 ratio

An Analogy

- You are trying to catch a fish from a pond. And you only have a net, then what would you do?
- Will you randomly throw your net?
- No, you will actually wait until you see one fish swimming around, then you would throw the net in that direction to collect the entire school.
- Therefore even if they are correlated, we still want to look at their entire group.

Elastic Net Regression

- Elastic regression works in a similar way.
- Let’ say, we have a bunch of correlated independent variables in a dataset, then elastic net will simply form a group consisting of these correlated variables. Now if any one of the variable of this group is a strong predictor (meaning having a strong relationship with dependent variable), then we will include the entire group in the model building, because omitting other variables (like what we did in lasso) might result in losing some information in terms of interpretation ability, leading to a poor model performance.

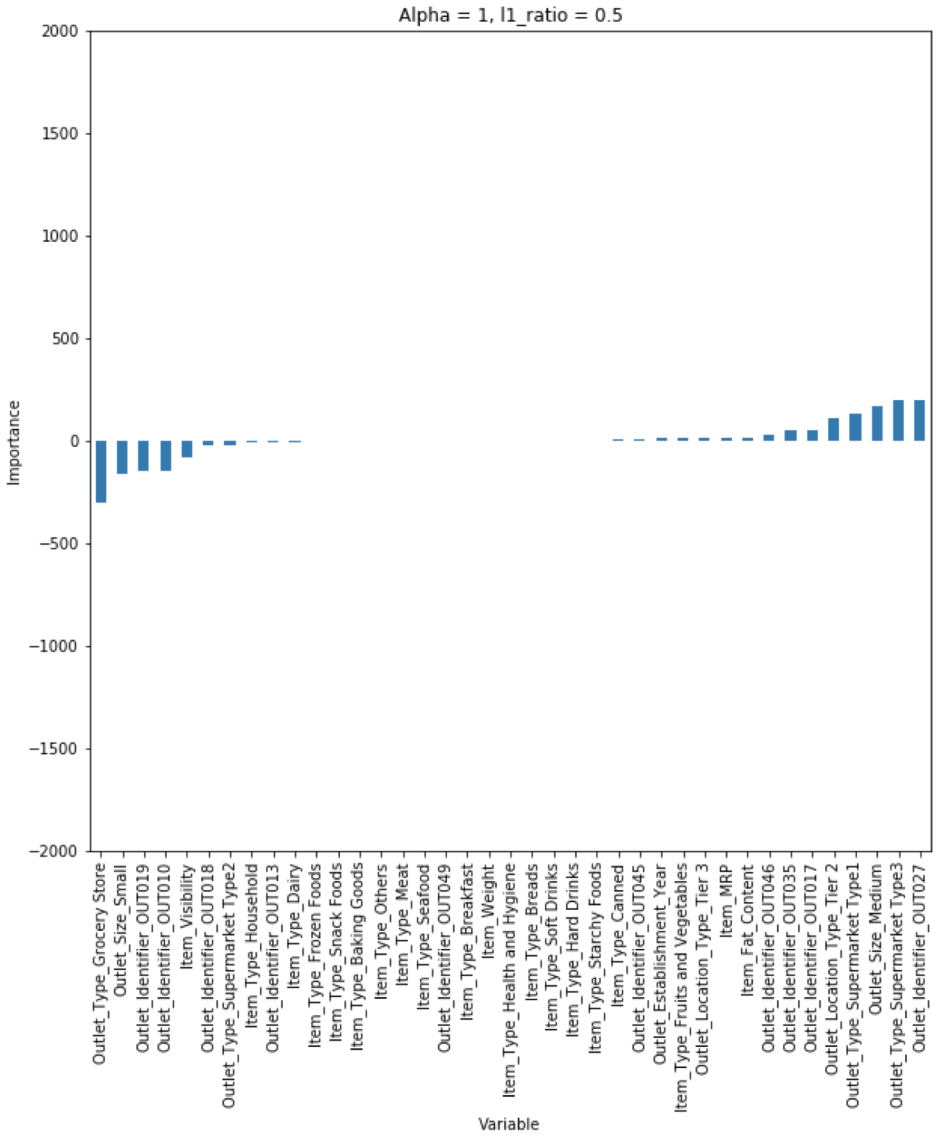
Tuning the Hyper-parameters

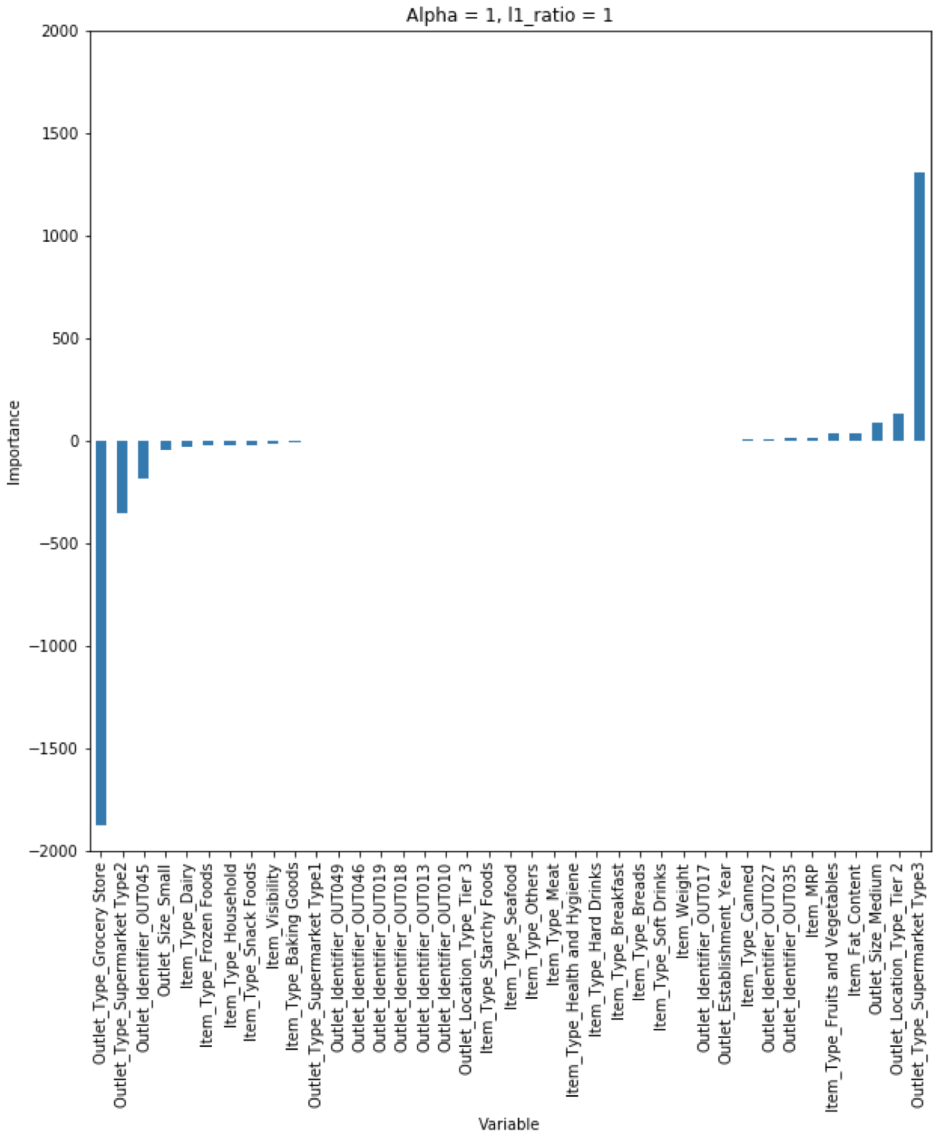
- The code contains two parameters: alpha and l1_ratio
- We can control the influence of Ridge and Lasso by adjusting these
- If λ1 and λ2 are the weights assigned to Lasso (L1) and Ridge (L2) respectively, then
- Alpha = λ1 + λ2
- l1_ratio = λ1 /(λ1 + λ2)

Tradeoff

Let alpha (or $\lambda 1 + \lambda 2$) = 1, and now consider the following cases:

- If l1_ratio =1, then from the formula of l1_ratio, we can see that l1_ratio can be 1 only if a=1, which implies b = 0. Therefore, it will be a lasso penalty.
- Similarly if l1_ratio = 0, implies a = 0. Then the penalty will be a ridge penalty.
- For l1_ratio between 0 and 1, the penalty is the combination of ridge and lasso.
- See two example runs on the next two slides


Alpha = 1, l1_ratio = 0.5
Importance
2000
1500
1000
500
0
−500
−1000
−1500
−2000
Outlet_Type_Grocery Store
Outlet_Size_Small
Outlet_Identifier_OUT019
Outlet_Identifier_OUT010
Item_Visibility
Outlet_Identifier_OUT018
Outlet_Type_Supermarket Type2
Item_Type_Household
Outlet_Identifier_OUT013
Item_Type_Dairy
Item_Type_Frozen Foods
Item_Type_Snack Foods
Item_Type_Baking Goods
Item_Type_Others
Item_Type_Meat
Item_Type_Seafood
Outlet_Identifier_OUT049
Item_Type_Breakfast
Item_Weight
Item_Type_Health and Hygiene
Item_Type_Breads
Item_Type_Soft Drinks
Item_Type_Hard Drinks
Item_Type_Starchy Foods
Item_Type_Canned
Outlet_Identifier_OUT045
Outlet_Establishment_Year
Item_Type_Fruits and Vegetables
Outlet_Location_Type_Tier 3
Item_MRP
Item_Fat_Content
Outlet_Identifier_OUT046
Outlet_Identifier_OUT035
Outlet_Identifier_OUT017
Outlet_Location_Type_Tier 2
Outlet_Type_Supermarket Type1
Outlet_Size_Medium
Outlet_Type_Supermarket Type3
Outlet_Identifier_OUT027
Variable


Alpha = 1, l1_ratio = 1
Importance
Variable
2000
1500
1000
500
0
−500
−1000
−1500
−2000
Outlet_Type_Grocery Store
Outlet_Type_Supermarket Type2
Outlet_Identifier_OUT045
Outlet_Size_Small
Item_Type_Dairy
Item_Type_Frozen Foods
Item_Type_Household
Item_Type_Snack Foods
Item_Visibility
Item_Type_Baking Goods
Outlet_Type_Supermarket Type1
Outlet_Identifier_OUT049
Outlet_Identifier_OUT046
Outlet_Identifier_OUT019
Outlet_Identifier_OUT018
Outlet_Identifier_OUT013
Outlet_Identifier_OUT010
Outlet_Location_Type_Tier 3
Item_Type_Starchy Foods
Item_Type_Seafood
Item_Type_Others
Item_Type_Meat
Item_Type_Health and Hygiene
Item_Type_Hard Drinks
Item_Type_Breakfast
Item_Type_Breads
Item_Type_Soft Drinks
Item_Weight
Outlet_Identifier_OUT017
Outlet_Establishment_Year
Item_Type_Canned
Outlet_Identifier_OUT027
Outlet_Identifier_OUT035
Item_MRP
Item_Type_Fruits and Vegetables
Item_Fat_Content
Outlet_Size_Medium
Outlet_Location_Type_Tier 2
Outlet_Type_Supermarket Type3