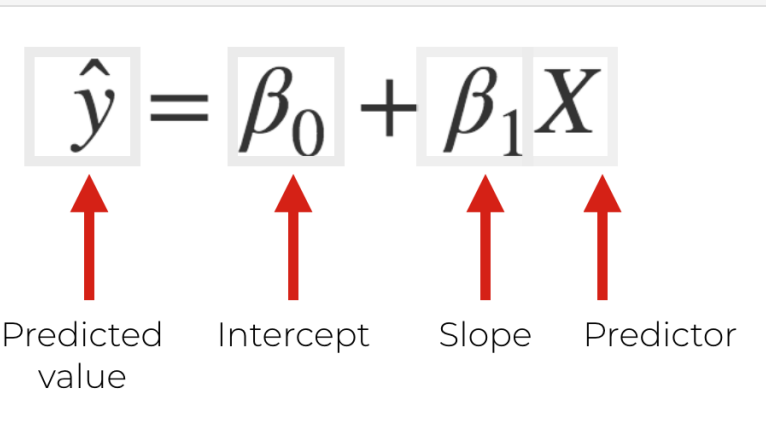
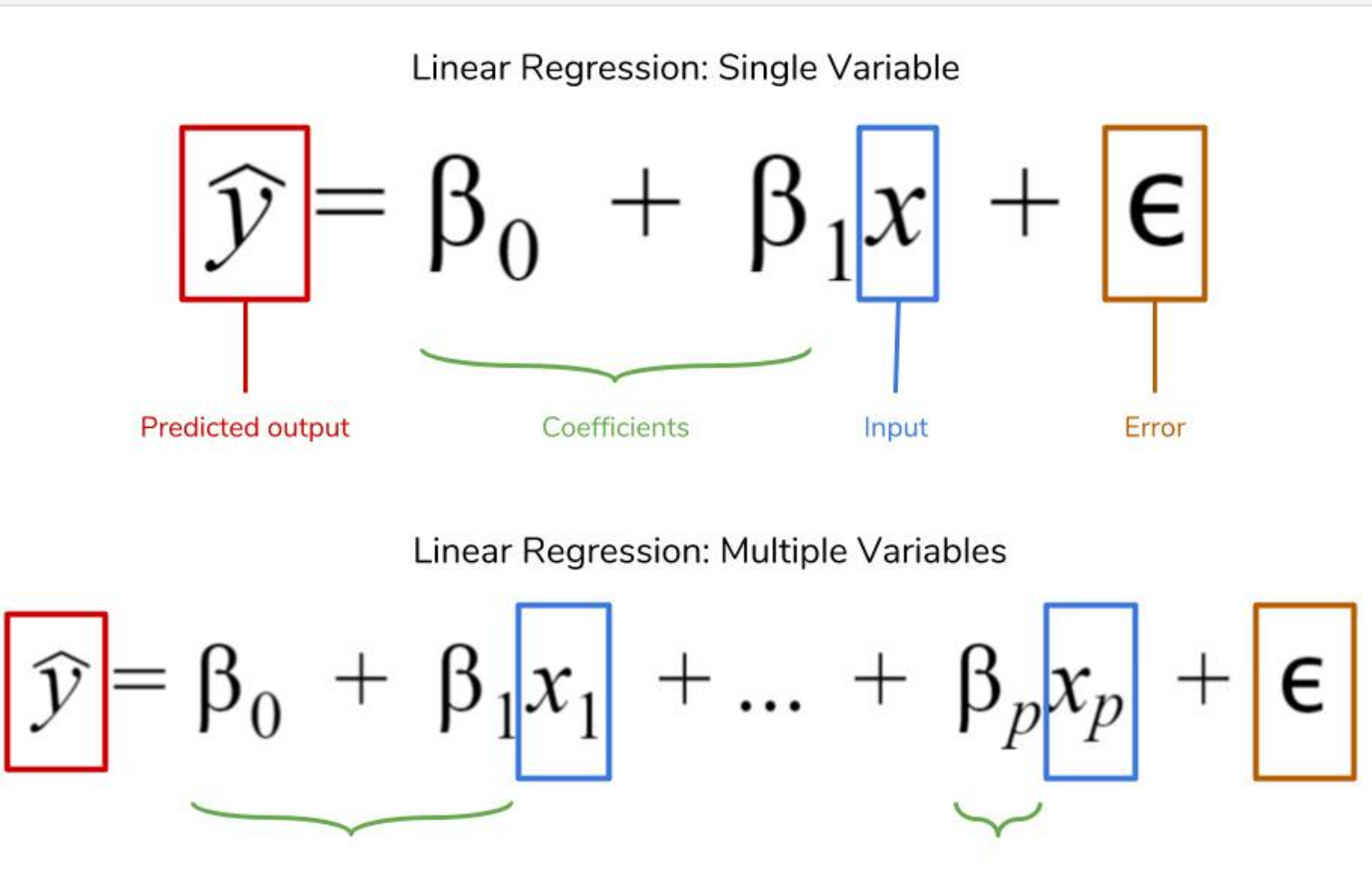
Use Of Linear and Logistics Regression Coefficients with Lasso (L1) and Ridge (L2) Regularization for Feature Selection In Machine Learning.

Linear Regression Equation.

```
from IPython.display import Image
Image(filename='C:/Users/Microsoft/Desktop/pandas/linear-regression-equation-explained.png',height=400,width=40
```




```
from IPython.display import Image
Image(filename='C:/Users/Microsoft/Desktop/pandas/DT4H1Yk.jpg',height=800,width=800)
```



- Linear regression is a straight foreward approach for predicting the quantitive response via on the basis on the different predicrive variables of X1 and X2 etc.
- It assumes that there is Linear Relationship between the x and y.
- That's means the by addition and subtraction y can be predicted based on the input features those input are x1,x2 we can say them x vector.
- βo,β1 and βp are the coefficient of linear regression.
- ε(Epsilon) is the error.:-This error having many types so we can as it is it's an error.There are diffrent types of the error which get introduces to a model we train ,test our model.

Why we use Linear Regression ?

- While working on the linear regreesion we have to make some basic assumptions those basic assuptions are like.
- We can predict y by using the linear combinations of the input vectors and feature space x should have **guassian distribution.** and **features** are **not correlated with each other that means there should not be any multi-colinearity.**
- That means the feature space won't be correcetd between and themselves.
- Features should be into the same scale x1..xn that means we need to do **Normalization or Standardization** bring this feature into a same scale.
- Since,In this lesson we are talking how can get the feature selection based on the coefficient of the linear and logistics regression.
- This coefficient of regression we will calculate by the **LASSO and Ridge Regularization**

What is the Regulirazation ?

- Regulirazation is the technique to **dicourage the complexity of the model**.It does by the **penalizing the loss function.** It helps to **solve the overfitting problem into our machine learning model** and it will improve the accuracy and reduce the complexity of the model and It increase the interpretability of the model.
- It adds the penality on the diffrent parameter of model to reduce the freedom of the model that means to reduce the complexity of model.
- Hence the model will be the less likely to fit the noise of the training data and it will improve the generalization ability of the model.

- `L1 Regulirazation` (also called Lasso)
- `L2 Regulirazation` (also called Ridge)
- `L1/L2 Regulirazation` (also called Elastic Net)

- `Lasso is help to shrike the coefficient which are less important to the zero.That means with the help of lasso regulirazation we can remove a few features.`
- `Ridge regression it doesn't reduce the coeficient to zero but it reduce the regression coefficient and with the reduction of the coefficient we can identify which feature have important in final prediction.`

For an exmaple:-

- Here, `β1` is the linear or logistice regression coefient and `x1 and xp` are the independent features.
- If we have the `β1` is more than the βp that time `x1` is more important than the `xp`.
- A regression model that uses L1 regulirazation Techniques is called as LASSO regression.
- A regression model that uses L2 regulirazation Techniques is called as Ridge regression.

What Is Lasso Regulirization ?

- There are three sources of an error.
- That's means we having three types of errors by which the error inroduce by testing and training model.
  - Noise
  - Bias
  - Variance
- We can't do anything about the noise but somehow we can do about the Bias and Variance.

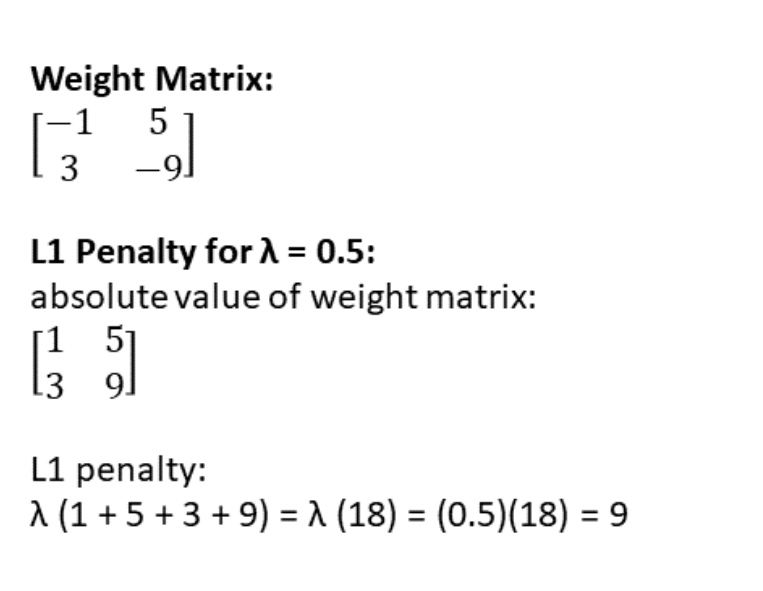
Trade Off inbetween the Bias-Variance.

We need to found such sweet spot on that spot we will minimum error and optimal complexity and minimum variance and baise there will found the optimal complexity.

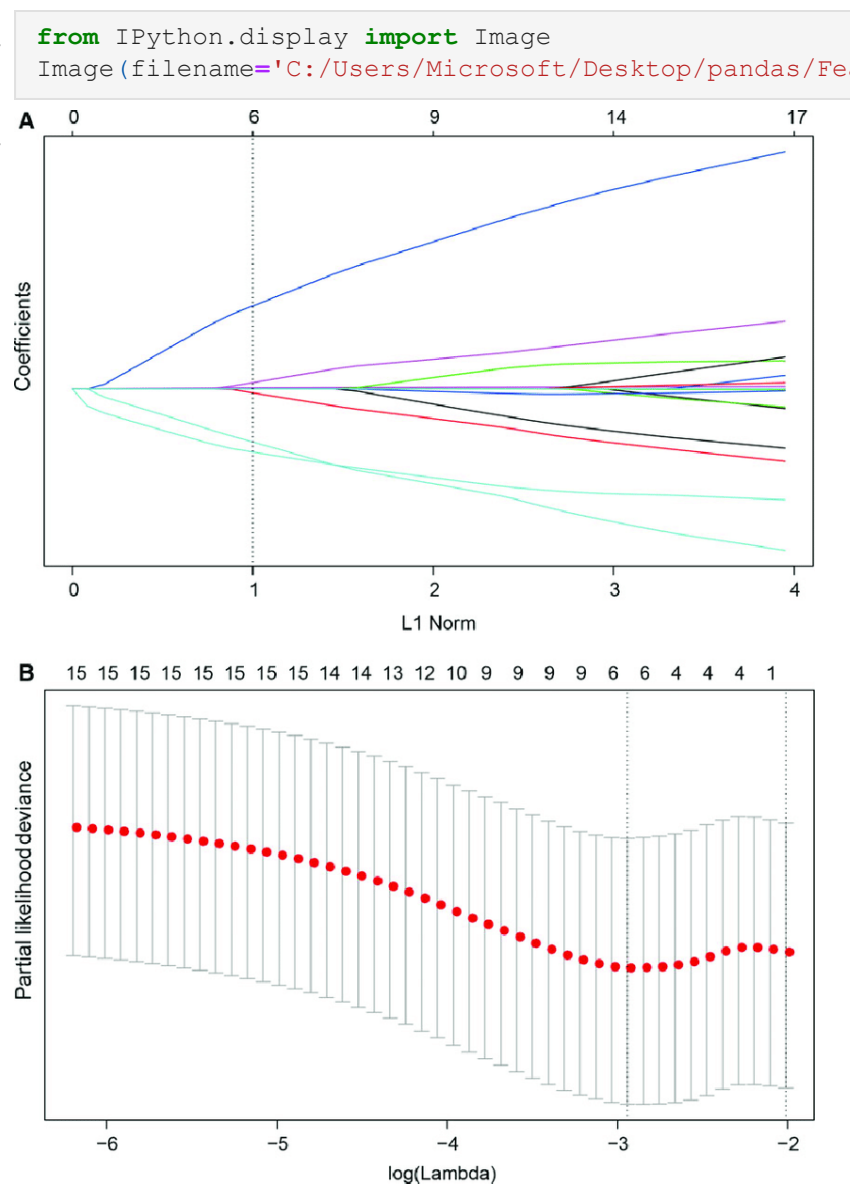
Logic And Mathematics Behind The Losso Regression.

- Lasso regulirization adds panality equivalent to absolute sum of regression coeficients values.
- In the equations:-
  - `Error Function(RSS)` :- Loss function
  - `λ` :- Regularization Coeficient or Optimization Coefficient.
- In this what we are doing we add some panlality into a loss functions.
- Now we have to find the Required sweet spot where the generization error shold be the less and overall ingeneral model perform better.
- By using the L1 over the regression coeficient few parameter will become zero.Whichever the feature doesn't much information to identify the output.
- `RSS` :- The residual sum of squares (RSS) is a statistical technique used to measure the amount of variance in a data set that is not explained by a regression model itself. Instead, it estimates the variance in the residuals, or error term.
- Equation = `RSS(w) + λ||w||`
  - `RSS` is the loss function is the error function or residual sum square.
  - `λ` is the Regularization Coeficient or Optimization Coefficient or panalty that Lasso regression add.
  - `w` is absolute sum of the regression coeficients.

```
from IPython.display import Image
Image(filename='C:/Users/Microsoft/Desktop/pandas/regularization-l1example.png',height=400,width=400)
```




```
from IPython.display import Image
Image(filename='C:/Users/Microsoft/Desktop/pandas/Feature-selection-using-LASSO-regression-model-A-Tuning-param
```



How we can choose Best Lambda (λ) ?

- Here `λ` is nothing but `L1` Reularization Coefficient.
- By doing the `Cross-Validation set` over the data we will get the best lambda value.

- `Below we having the steps.`
  - Suppose We have sufficient amount the data .
  - We can split our data into the training set validations and testing set.
  - In the training set we fit our model then we will get our regression coefficient with the help of regularization.
  - Then we test model performace on the validation set
  - If there is something which is not going give us good accuracy or something wrong in there so in order to detect that we have to validate the our model.validate on the validation set.There we will find something which is good to go ahead then we changes the parameter goes back to the training set and do the optimization on the validation set.
  - Finally our features will do the generised testing on the test set and this is how we choose the best and optimized lambda during the our regulizations.

What Is Ridge Regulirization ?

- Ridge regulirization is also know as `L2 regularizations` which adds penalty is equal to the `sum of squared value of the coeficients of regression.`
- It we adds the penalties on the **sum of Squared regression coefficient**.
- Here the `λ` i.e.L2 Regulirization Coeficient works as Tunning Parameter.

Comparision with RSS(Sum of Squared),Lambda AND Weight.

- `Equation = RSS(w) + square(λ||w||)`

Lambda is High :-

- L2 Reulirization is high that is Lambda and that time the result we will get the biased somewhere else.
- That means in this case we will get the high bias low variance.
- Under the assumption If lambda is increased upto the infinity that time the `w` became so small almost zero i.e.0.

Lambda Is Low :-

- L2 regulirization is low that is Lambda and that time the result face the with number of features because of the L2 Regulirization coeficient is low and it adds the less impact over the data.
- Due to this it will end up the overfitting of the due to high dimensional i.e.feature training.
- That will confuse the model.
- Model will train the add point so there will be the high variance low bias.

- λ is the tunning parameter which is performing perfect in our case.
- We can say the lambda is the strength of the regularizations.
- L2 paramenter we will force the parameter to be relatively small,the bigger panalization (and the more robust) the coeficient are.

Diffrence Between the L1 and L2 Regulirization

L1 Regulirization:-

- L1 penalizes the sum of absolutes value of weights.
- L1 has spare solutions.
- L1 has multiple solutions.
- L1 has built in feature selection.
- L1 is robust to outliers.
- L1 generates a model that are simple and interpretable but can not learn the complex patterns.

L2 Regulirization:-

- L2 Regulizations penalizes the sum of square weights.
- L2 has non spare solutions.
- L2 has one solution.
- L2 has no feature selections.
- L2 has not robust to outliers.
- L2 gives better predictions even output variable is a function of all input features.
- L2 regularization is able to learn complex data patterns.

Start With Machine Learning.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

```
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectKBest
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
```

```
from sklearn.linear_model import LinearRegression,LogisticRegression
from sklearn.feature_selection import SelectFromModel
```

```
data = pd.read_csv('Data_train_reduced.csv')
```

```
pd.pandas.set_option('display.max_columns',None)
```

```
data.head()
```

	Respondent.ID	Product.ID	Product	Instant.Liking	q1_1.personal.opinion.of.this.Deodorant	q2_all.words	q3_1.strength.of.the.Deodorant	q
0	3800	121	Deodorant B	1	4	1	4	
1	3801	121	Deodorant B	0	5	1	4	
2	3802	121	Deodorant B	0	6	1	3	
3	3803	121	Deodorant B	1	4	0	4	
4	3804	121	Deodorant B	1	4	1	2	

```
data.columns
```

```
Index(['Respondent.ID', 'Product.ID', 'Product', 'Instant.Liking',
       'q1_1.personal.opinion.of.this.Deodorant', 'q2_all.words',
       'q3_1.strength.of.the.Deodorant', 'q4_1.artificial.chemical',
       'q4_2.attractive', 'q4_3.bold', 'q4_4.boring', 'q4_5.casual',
       'q4_6.cheap', 'q4_7.clean', 'q4_8.easy.to.wear', 'q4_9.elegant',
       'q4_10.feminine', 'q4_11.for.someone.like.me', 'q4_12.heavy',
       'q4_13.high.quality', 'q4_14.long.lasting', 'q4_15.masculine',
       'q4_16.memorable', 'q4_17.natural', 'q4_18.old.fashioned',
       'q4_19.ordinary', 'q4_20.overpowering', 'q4_21.sharp',
       'q4_22.sophisticated', 'q4_23.upscale', 'q4_24.well.rounded',
       'q5_1.Deodorant.is.addictive', 'q7', 'q8.1', 'q8.2', 'q8.5', 'q8.6',
       'q8.7', 'q8.8', 'q8.9', 'q8.10', 'q8.11', 'q8.12', 'q8.13', 'q8.17',
       'q8.18', 'q8.19', 'q8.20',
       'q9.how.likely.would.you.be.to.purchase.this.Deodorant',
       'q10.prefer.this.Deodorant.or.your.usual.Deodorant',
       'q11.time.of.day.would.this.Deodorant.be.appropriate',
       'q12.which.occasions.would.this.Deodorant.be.appropriate',
       'Q13_Liking.after.30.minutes',
       'q14.Deodorant.overall.on.a.scale.from.1.to.10', 'ValSegb',
       's7.involved.in.the.selection.of.the.cosmetic.products',
       's8.ethnic.background', 's9.education', 's10.income',
       's11.marital.status', 's12.working.status', 's13.2',
       's13a.b.most.often', 's13b.bottles.of.Deodorant.do.you.currently.own'],
      dtype='object')
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2500 entries, 0 to 2499
Data columns (total 64 columns):
Respondent.ID                                              2500 non-null int64
Product.ID                                                 2500 non-null int64
Product                                                    2500 non-null object
Instant.Liking                                             2500 non-null int64
q1_1.personal.opinion.of.this.Deodorant                    2500 non-null int64
q2_all.words                                               2500 non-null int64
q3_1.strength.of.the.Deodorant                             2500 non-null int64
q4_1.artificial.chemical                                   2500 non-null int64
q4_2.attractive                                            2500 non-null int64
q4_3.bold                                                  2500 non-null int64
q4_4.boring                                                2500 non-null int64
q4_5.casual                                                2500 non-null int64
q4_6.cheap                                                 2500 non-null int64
q4_7.clean                                                 2500 non-null int64
q4_8.easy.to.wear                                          2500 non-null int64
q4_9.elegant                                               2500 non-null int64
q4_10.feminine                                             2500 non-null int64
q4_11.for.someone.like.me                                  2500 non-null int64
q4_12.heavy                                                2500 non-null int64
q4_13.high.quality                                         2500 non-null int64
q4_14.long.lasting                                         2500 non-null int64
q4_15.masculine                                            2500 non-null int64
q4_16.memorable                                            2500 non-null int64
q4_17.natural                                              2500 non-null int64
q4_18.old.fashioned                                        2500 non-null int64
q4_19.ordinary                                             2500 non-null int64
q4_20.overpowering                                         2500 non-null int64
q4_21.sharp                                                2500 non-null int64
q4_22.sophisticated                                        2500 non-null int64
q4_23.upscale                                              2500 non-null int64
q4_24.well.rounded                                         2500 non-null int64
q5_1.Deodorant.is.addictive                                2500 non-null int64
q7                                                         2500 non-null int64
q8.1                                                       2500 non-null float64
q8.2                                                       2000 non-null float64
q8.5                                                       2500 non-null int64
q8.6                                                       2500 non-null int64
q8.7                                                       2000 non-null float64
q8.8                                                       1500 non-null float64
q8.9                                                       500 non-null float64
q8.10                                                      500 non-null float64
q8.11                                                      2500 non-null int64
q8.12                                                      2000 non-null float64
q8.13                                                      2500 non-null int64
q8.17                                                      500 non-null float64
q8.18                                                      500 non-null float64
q8.19                                                      2500 non-null int64
q8.20                                                      1000 non-null float64
q9.how.likely.would.you.be.to.purchase.this.Deodorant      2500 non-null int64
q10.prefer.this.Deodorant.or.your.usual.Deodorant          2500 non-null int64
q11.time.of.day.would.this.Deodorant.be.appropriate        2500 non-null int64
q12.which.occasions.would.this.Deodorant.be.appropriate    2500 non-null int64
Q13_Liking.after.30.minutes                                2500 non-null int64
q14.Deodorant.overall.on.a.scale.from.1.to.10              2500 non-null int64
ValSegb                                                    2500 non-null int64
s7.involved.in.the.selection.of.the.cosmetic.products      2500 non-null int64
s8.ethnic.background                                       2500 non-null int64
s9.education                                               2500 non-null int64
s10.income                                                 2500 non-null int64
s11.marital.status                                         2500 non-null int64
s12.working.status                                         2500 non-null int64
s13.2                                                      2500 non-null int64
s13a.b.most.often                                          2500 non-null int64
s13b.bottles.of.Deodorant.do.you.currently.own             2500 non-null int64
dtypes: float64(10), int64(53), object(1)
memory usage: 1.2+ MB
```

```
data.isnull().sum().sort_values(ascending=False).head(10)
```

```
q8.10                        2000
q8.9                         2000
q8.17                        2000
q8.18                        2000
q8.20                        1500
q8.8                         1000
q8.2                          500
q8.7                          500
q8.12                         500
q4_11.for.someone.like.me       0
dtype: int64
```

```
data['Instant.Liking'].value_counts()
```

```
0    1882
1     618
Name: Instant.Liking, dtype: int64
```

```
data.drop(columns=['q8.10','q8.9','q8.8','q8.17','q8.18','q8.20','q8.2'],inplace=True)
```

```
data['q8.7'].mean()
```

```
0.0415
```

```
data['q8.7'].median()
```

```
0.0
```

```
data['q8.7'].fillna(0.0,inplace=True)
```

```
data['q8.12'].mean()
```

```
0.0625
```

```
data['q8.12'].median()
```

```
0.0
```

```
data['q8.12'].fillna(0.0,inplace=True)
```

```
data.isnull().sum()
```

```
Respondent.ID                                              0
Product.ID                                                 0
Product                                                    0
Instant.Liking                                             0
q1_1.personal.opinion.of.this.Deodorant                    0
q2_all.words                                               0
q3_1.strength.of.the.Deodorant                             0
q4_1.artificial.chemical                                   0
q4_2.attractive                                            0
q4_3.bold                                                  0
q4_4.boring                                                0
q4_5.casual                                                0
q4_6.cheap                                                 0
q4_7.clean                                                 0
q4_8.easy.to.wear                                          0
q4_9.elegant                                               0
q4_10.feminine                                             0
q4_11.for.someone.like.me                                  0
q4_12.heavy                                                0
q4_13.high.quality                                         0
q4_14.long.lasting                                         0
q4_15.masculine                                            0
q4_16.memorable                                            0
q4_17.natural                                              0
q4_18.old.fashioned                                        0
q4_19.ordinary                                             0
q4_20.overpowering                                         0
q4_21.sharp                                                0
q4_22.sophisticated                                        0
q4_23.upscale                                              0
q4_24.well.rounded                                         0
q5_1.Deodorant.is.addictive                                0
q7                                                         0
q8.1                                                       0
q8.5                                                       0
q8.6                                                       0
q8.7                                                       0
q8.11                                                      0
q8.12                                                      0
q8.13                                                      0
q8.19                                                      0
q9.how.likely.would.you.be.to.purchase.this.Deodorant      0
q10.prefer.this.Deodorant.or.your.usual.Deodorant          0
q11.time.of.day.would.this.Deodorant.be.appropriate        0
q12.which.occasions.would.this.Deodorant.be.appropriate    0
Q13_Liking.after.30.minutes                                0
q14.Deodorant.overall.on.a.scale.from.1.to.10              0
ValSegb                                                    0
s7.involved.in.the.selection.of.the.cosmetic.products      0
s8.ethnic.background                                       0
s9.education                                               0
s10.income                                                 0
s11.marital.status                                         0
s12.working.status                                         0
s13.2                                                      0
s13a.b.most.often                                          0
s13b.bottles.of.Deodorant.do.you.currently.own             0
dtype: int64
```

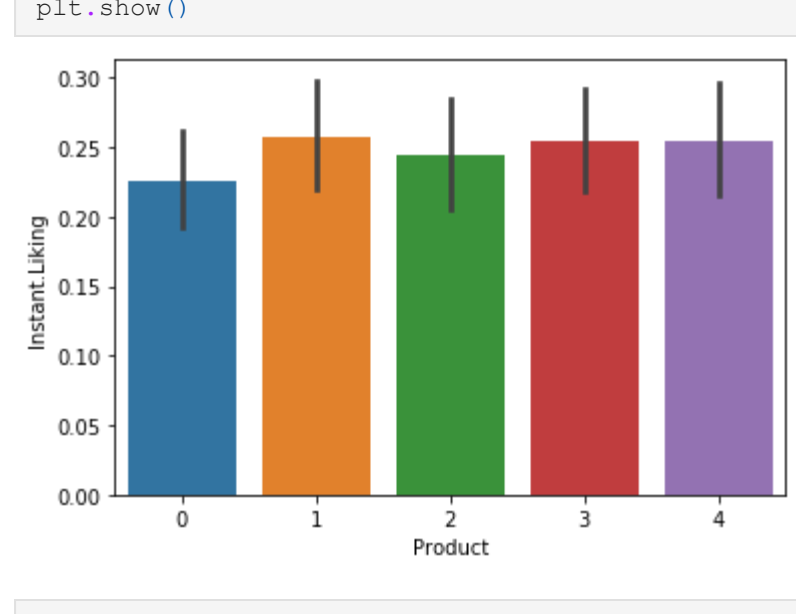
```
data.drop(columns=['Respondent.ID','Product.ID'],inplace=True)
```

```
data['Product'].value_counts()
```

```
Deodorant H    500
Deodorant G    500
Deodorant B    500
Deodorant J    500
Deodorant F    500
Name: Product, dtype: int64
```

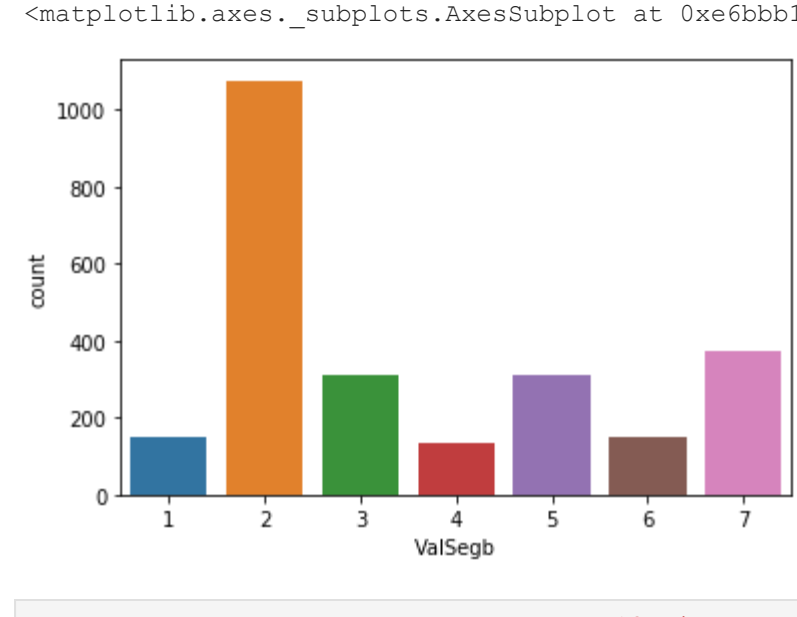
```
x= {'Deodorant H':0,'Deodorant J':1,'Deodorant G':2,'Deodorant B':3,'Deodorant F':4}
data['Product']=data['Product'].map(x)
```

```
sns.barplot(data['Product'],data['Instant.Liking'])
plt.show()
```

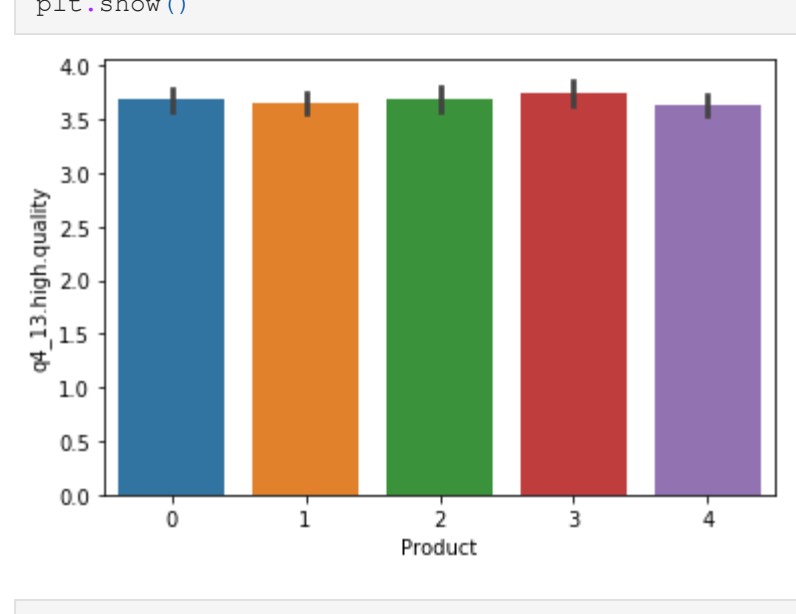



```
sns.countplot(data['ValSegb'])
```

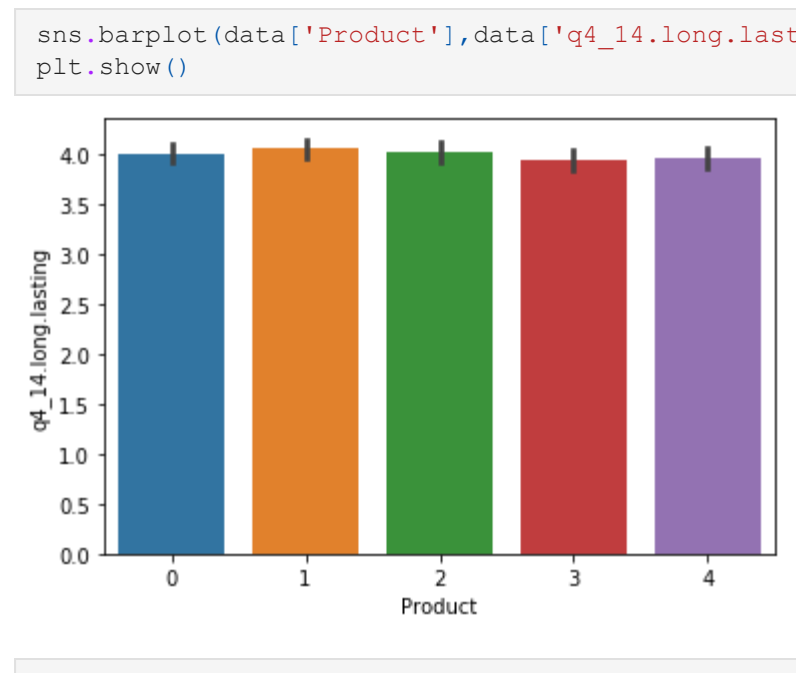
```
<matplotlib.axes._subplots.AxesSubplot at 0xe6bb10>
```



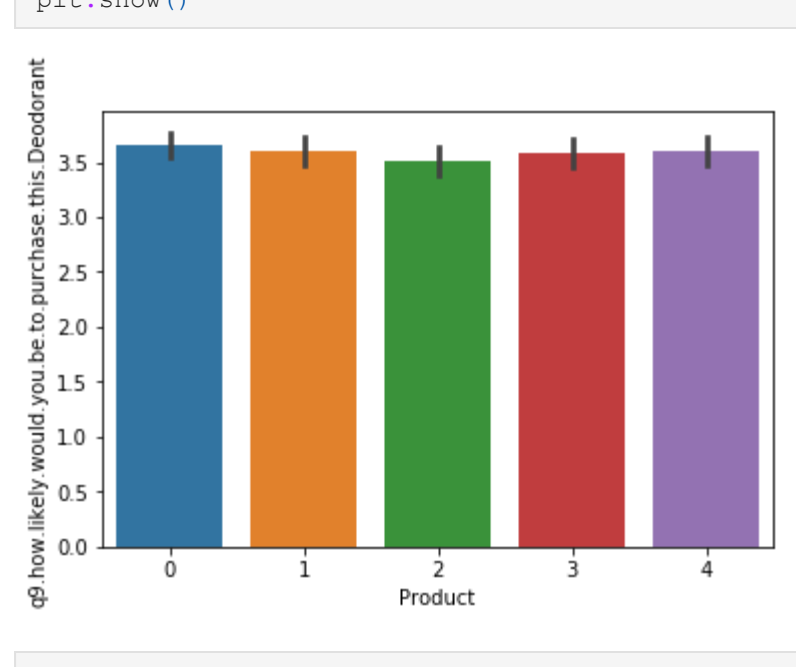

```
sns.barplot(data['Product'],data['q4_13.high.quality'])
plt.show()
```




```
sns.barplot(data['Product'],data['q4_14.long.lasting'])
plt.show()
```




```
sns.barplot(data['Product'],data['q9.how.likely.would.you.be.to.purchase.this.Deodorant'])
plt.show()
```




```
plt.figure(figsize=(20,15))
data.hist(figsize=(20,20))
plt.tight_layout()
plt.grid()
plt.show()
```

```
<Figure size 1440x1080 with 0 Axes>
```

```
x = data.drop(columns=['Instant.Liking'])
y = data['Instant.Liking']
```

```
y.value_counts()
```

```
0    1882
1     618
Name: Instant.Liking, dtype: int64
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y,train_size=0.80,random_state=42)
```

```
x_train.shape,x_test.shape
```

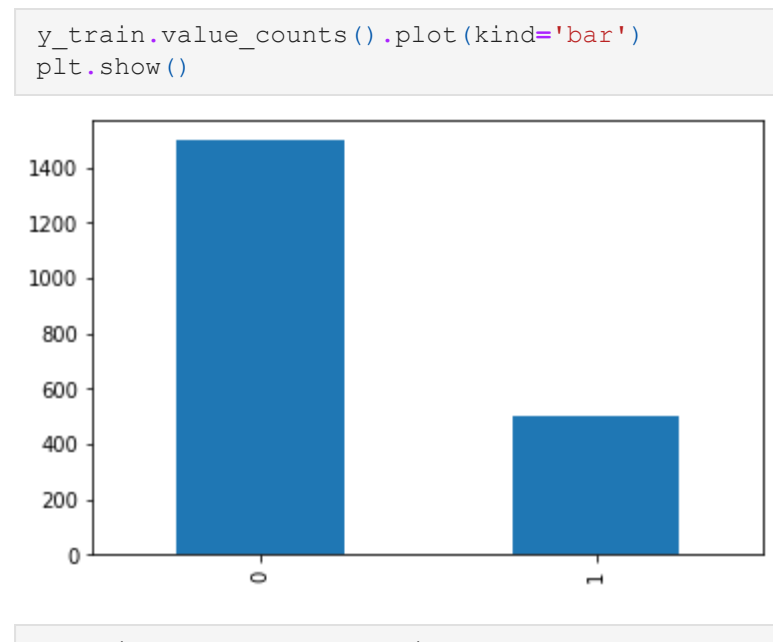
```
((2000, 54), (500, 54))
```

```
y_train.value_counts()
```

```
0    1497
1     503
Name: Instant.Liking, dtype: int64
```

Make the imbalance to Balace datasets.

```
y_train.value_counts().plot(kind='bar')
plt.show()
```



```
from imblearn.over_sampling import SMOTE
```

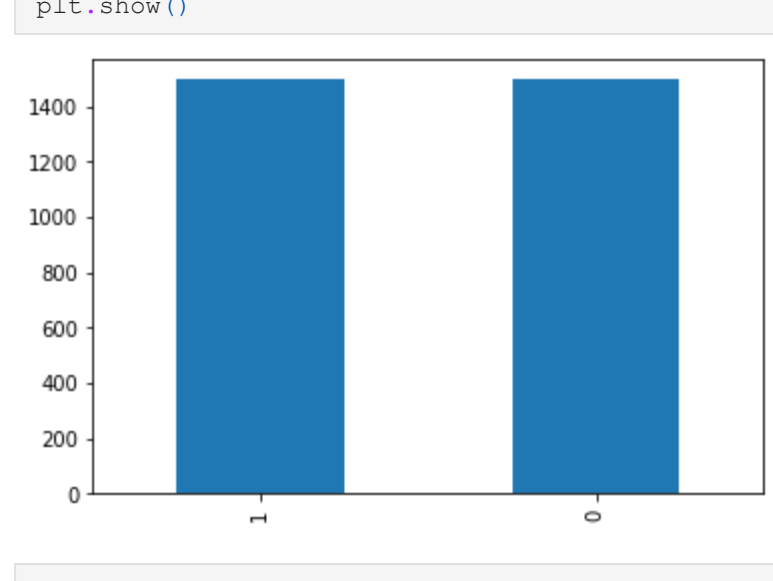
```
sm = SMOTE(sampling_strategy=1,random_state=42,k_neighbors=5)
```

```
x_train_res,y_train_res = sm.fit_sample(x_train,y_train)
```

```
y_train_res.value_counts()
```

```
1    1497
0    1497
Name: Instant.Liking, dtype: int64
```

```
y_train_res.value_counts().plot(kind='bar')
plt.show()
```



```
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
X_train= scaler.fit_transform(x_train_res)
X_test= scaler.transform(x_test)
```

L1 Reularization By using the Logistics Regression

- l1 Reularization is the lasso Regression.
- C is inverse of the strengh of the Regularization.

```
sel = SelectFromModel(LogisticRegression(penalty='l1',C=80,solver='liblinear'))
```

```
sel.fit(X_train,y_train_res)
```

```
SelectFromModel(estimator=LogisticRegression(C=80, class_weight=None,
                                             dual=False, fit_intercept=True,
                                             intercept_scaling=1, l1_ratio=None,
                                             max_iter=100, multi_class='auto',
                                             n_jobs=None, penalty='l1',
                                             random_state=None,
                                             solver='liblinear', tol=0.0001,
                                             verbose=0, warm_start=False),
                max_features=None, norm_order=1, prefit=False, threshold=None)
```

```
sel.estimator_.coef_
```

```
array([[ 0.00000000e+00, -3.01453630e+01, -1.24991230e-01,
         0.00000000e+00, -3.52988216e-02, -4.50633706e-03,
        -6.49010440e-02, -1.11229114e-01,  0.00000000e+00,
        -8.06946648e-02,  0.00000000e+00, -3.59955392e-02,
        -1.88054172e-02,  0.00000000e+00, -5.26304286e-02,
         0.00000000e+00,  9.43427443e-03, -1.22243396e-02,
        -1.05561570e-01, -5.82560021e-02,  0.00000000e+00,
        -1.91356402e-02, -5.53210032e-02, -5.39530728e-02,
        -4.15197846e-02,  0.00000000e+00, -4.15172793e-02,
        -4.23809891e-02,  0.00000000e+00, -1.19124366e-01,
         0.00000000e+00, -4.65568780e-02, -9.03836288e-02,
         0.00000000e+00,  0.00000000e+00, -1.95613505e-03,
        -2.98050247e-02,  2.49975443e-02,  0.00000000e+00,
         0.00000000e+00, -7.63440353e-02, -1.09162827e-02,
        -8.95403945e-03, -2.96951855e-02, -8.66955147e-04,
         0.00000000e+00, -9.26274336e-03,  0.00000000e+00,
        -1.04356127e-02, -4.40616900e-02,  0.00000000e+00,
         0.00000000e+00, -5.02331464e-02, -8.40812337e-03]])
```

```
sel.get_support()
```

```
array([False,  True,  True, False,  True,  True,  True,  True, False,
        True, False,  True,  True, False,  True, False,  True,  True,
        True,  True, False,  True,  True,  True,  True, False,  True,
        True, False,  True, False,  True,  True, False, False,  True,
        True,  True, False, False,  True,  True,  True,  True,  True,
       False,  True, False,  True,  True, False, False,  True,  True])
```

```
features = x.columns[sel.get_support()]
features
```

```
Index(['q1_1.personal.opinion.of.this.Deodorant', 'q2_all.words',
       'q4_1.artificial.chemical', 'q4_2.attractive', 'q4_3.bold',
       'q4_4.boring', 'q4_6.cheap', 'q4_8.easy.to.wear', 'q4_9.elegant',
       'q4_11.for.someone.like.me', 'q4_13.high.quality', 'q4_14.long.lasting',
       'q4_15.masculine', 'q4_16.memorable', 'q4_18.old.fashioned',
       'q4_19.ordinary', 'q4_20.overpowering', 'q4_21.sharp', 'q4_23.upscale',
       'q4_24.well.rounded', 'q7', 'q8.5', 'q8.6', 'q8.12', 'q8.13', 'q8.19',
       'q11.time.of.day.would.this.Deodorant.be.appropriate',
       'q12.which.occasions.would.this.Deodorant.be.appropriate',
       'Q13_Liking.after.30.minutes',
       'q14.Deodorant.overall.on.a.scale.from.1.to.10', 'ValSegb',
       's8.ethnic.background', 's10.income', 's11.marital.status',
       's13a.b.most.often', 's13b.bottles.of.Deodorant.do.you.currently.own'],
      dtype='object')
```

```
x_train_rel = sel.transform(X_train)
x_test_rel  = sel.transform(X_test)
```

```
x_train_rel.shape,x_test_rel.shape
```

```
((2994, 36), (500, 36))
```

```
x_train.columns[sel.get_support()]
```

```
Out[112… Index(['q1_1.personal.opinion.of.this.Deodorant', 'q2_all.words',
       'q4_1.artificial.chemical', 'q4_2.attractive', 'q4_3.bold',
       'q4_4.boring', 'q4_6.cheap', 'q4_8.easy.to.wear', 'q4_9.elegant',
       'q4_11.for.someone.like.me', 'q4_13.high.quality', 'q4_14.long.lasting',
       'q4_15.masculine', 'q4_16.memorable', 'q4_18.old.fashioned',
       'q4_19.ordinary', 'q4_20.overpowering', 'q4_21.sharp', 'q4_23.upscale',
       'q4_24.well.rounded', 'q7', 'q8.5', 'q8.6', 'q8.12', 'q8.13', 'q8.19',
       'q11.time.of.day.would.this.Deodorant.be.appropriate',
       'q12.which.occasions.would.this.Deodorant.be.appropriate',
       'Q13_Liking.after.30.minutes',
       'q14.Deodorant.overall.on.a.scale.from.1.to.10', 'ValSegb',
       's8.ethnic.background', 's10.income', 's11.marital.status',
       's13a.b.most.often', 's13b.bottles.of.Deodorant.do.you.currently.own'],
      dtype='object')
```

```
In [113… from sklearn.ensemble import RandomForestClassifier
```

```
In [114… classifier = RandomForestClassifier(n_estimators=100,criterion='entropy')
         classifier.fit(X_train,y_train_res)
         Y_pred = classifier.predict(X_test)
         print('Accuaracy :',accuracy_score(y_test,Y_pred))

Accuaracy : 1.0
```

```
In [115… classifier = RandomForestClassifier(n_estimators=100,criterion='gini')
         classifier.fit(X_train,y_train_res)
         Y_predict = classifier.predict(X_test)
         print('Accuaracy :',accuracy_score(y_test,Y_predict))

Accuaracy : 1.0
```

```
In [116… print('Model score in training data :-',classifier.score(X_train,y_train_res))
         print('Model score in testing data :-',classifier.score(X_test,y_test))

Model score in training data :- 1.0
Model score in testing data :- 1.0
```

L2 Regularization.

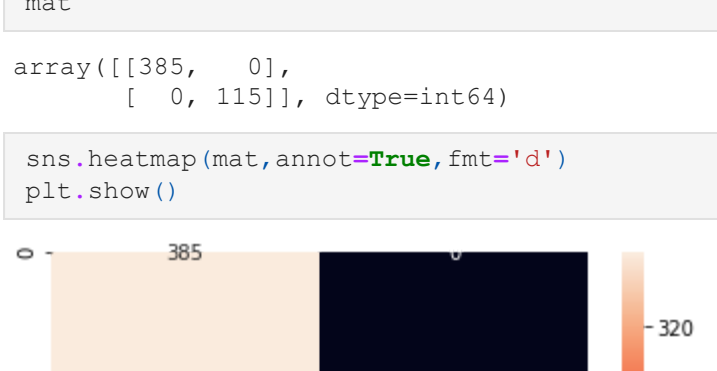
```
In [146… sel = SelectFromModel(LogisticRegression(penalty='l2',C=0.05,solver='liblinear'))
         sel.fit(x_train_res,y_train_res)

Out[146… SelectFromModel(estimator=LogisticRegression(C=0.05, class_weight=None,
                                                 dual=False, fit_intercept=True,
                                                 intercept_scaling=1, l1_ratio=None,
                                                 max_iter=100, multi_class='auto',
                                                 n_jobs=None, penalty='l2',
                                                 random_state=None,
                                                 solver='liblinear', tol=0.0001,
                                                 verbose=0, warm_start=False),
                max_features=None, norm_order=1, prefit=False, threshold=None)
```

```
In [151… x_train_l2 = sel.transform(x_train_res)
         x_test_l2  = sel.transform(x_test)
```

```
In [153… %%time
         classifier = RandomForestClassifier(n_estimators=100,criterion='gini')
         classifier.fit(x_train_l2,y_train_res)
         Y_pred = classifier.predict(x_test_l2)
         print('Accuaracy :',accuracy_score(y_test,Y_pred))

Accuaracy : 1.0
Wall time: 632 ms
```

```
In [154… mat = confusion_matrix(y_test,Y_pred)
         mat

Out[154… array([[385,   0],
                [  0, 115]], dtype=int64)
```

```
In [155… sns.heatmap(mat,annot=True,fmt='d')
         plt.show()
```




```
In [156… print(classification_report(y_test,Y_pred))

              precision    recall  f1-score   support

           0       1.00      1.00      1.00       385
           1       1.00      1.00      1.00       115

    accuracy                           1.00       500
   macro avg       1.00      1.00      1.00       500
weighted avg       1.00      1.00      1.00       500
```

Linear Regression Regularization.

```
In [126… data1 = pd.read_csv('kc_house_data.csv')
```

```
In [127… data1.head()
```

Out[127…

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	3	7	1180
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	3	7	2170
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	3	6	770
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	5	7	1050
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	3	8	1680

```
In [128… data1.tail()
```

Out[128…

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_ab
21608	263000018	20140521T000000	360000.0	3	2.50	1530	1131	3.0	0	0	3	8	1
21609	6600060120	20150223T000000	400000.0	4	2.50	2310	5813	2.0	0	0	3	8	2
21610	1523300141	20140623T000000	402101.0	2	0.75	1020	1350	2.0	0	0	3	7	1
21611	291310100	20150116T000000	400000.0	3	2.50	1600	2388	2.0	0	0	3	8	1
21612	1523300157	20141015T000000	325000.0	2	0.75	1020	1076	2.0	0	0	3	7	1

```
In [129… data1.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
id               21613 non-null int64
date             21613 non-null object
price            21613 non-null float64
bedrooms         21613 non-null int64
bathrooms        21613 non-null float64
sqft_living      21613 non-null int64
sqft_lot         21613 non-null int64
floors           21613 non-null float64
waterfront       21613 non-null int64
view             21613 non-null int64
condition        21613 non-null int64
grade            21613 non-null int64
sqft_above       21613 non-null int64
sqft_basement    21613 non-null int64
yr_built         21613 non-null int64
yr_renovated     21613 non-null int64
zipcode          21613 non-null int64
lat              21613 non-null float64
long             21613 non-null float64
sqft_living15    21613 non-null int64
sqft_lot15       21613 non-null int64
dtypes: float64(5), int64(15), object(1)
memory usage: 3.4+ MB
```

```
In [130… data.shape

Out[130… (2500, 55)
```

```
In [131… round(data.describe(),1)
```

Out[131…

	Product	Instant.Liking	q1_1.personal.opinion.of.this.Deodorant	q2_all.words	q3_1.strength.of.the.Deodorant	q4_1.artificial.chemical	q4_2
count	2500.0	2500.0	2500.0	2500.0	2500.0	2500.0	
mean	2.0	0.2	5.1	1.1	3.3	2.5	
std	1.4	0.4	1.5	0.9	0.7	1.4	
min	0.0	0.0	1.0	0.0	1.0	1.0	
25%	1.0	0.0	5.0	0.0	3.0	1.0	
50%	2.0	0.0	5.0	1.0	3.0	2.0	
75%	3.0	0.0	6.0	2.0	4.0	4.0	
max	4.0	1.0	7.0	5.0	5.0	5.0	

```
In [132… data1.isnull().sum()

Out[132… id               0
         date             0
         price            0
         bedrooms         0
         bathrooms        0
         sqft_living      0
         sqft_lot         0
         floors           0
         waterfront       0
         view             0
         condition        0
         grade            0
         sqft_above       0
         sqft_basement    0
         yr_built         0
         yr_renovated     0
         zipcode          0
         lat              0
         long             0
         sqft_living15    0
         sqft_lot15       0
         dtype: int64
```

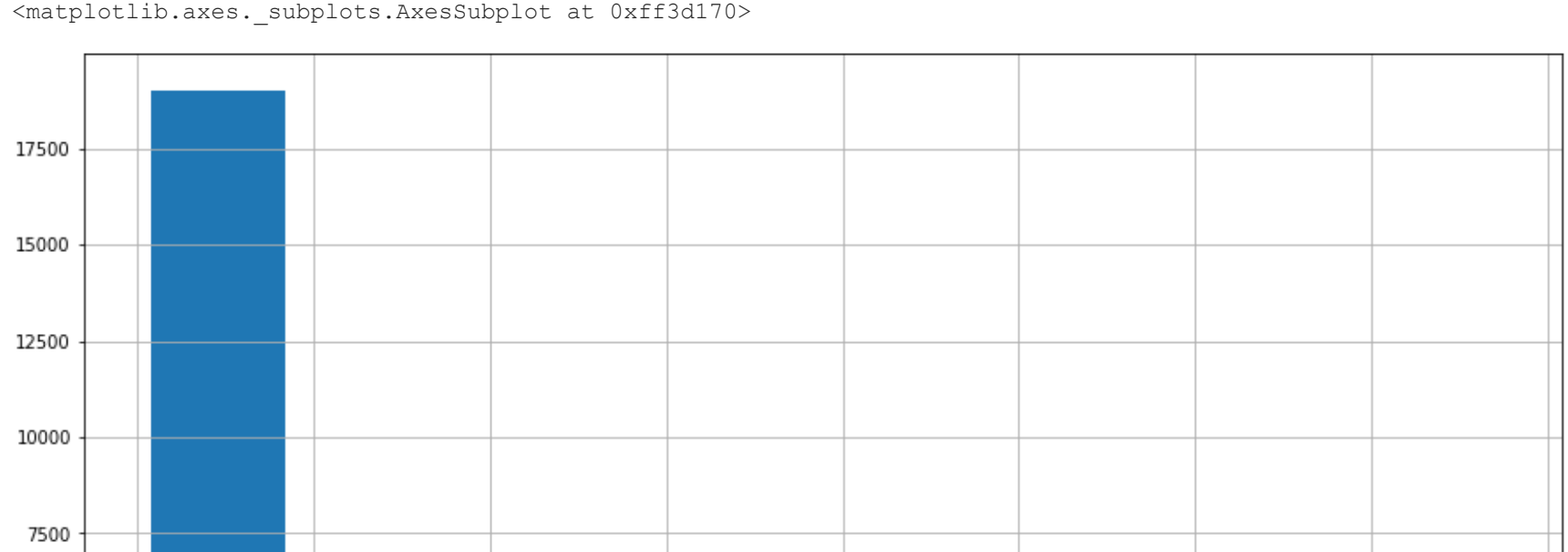
```
In [133… len(data1.columns)

Out[133… 21
```

```
In [134… data1.drop(columns=['id','date','zipcode'],inplace=True)
```

```
In [135… data1['yr_renovated'].value_counts().sort_values(ascending=False).head()

Out[135… 0       20699
         2014       91
         2013       37
         2003       36
         2000       35
         Name: yr_renovated, dtype: int64
```

```
In [136… data1['yr_built'].value_counts().sort_values(ascending=False).head()

Out[136… 2014    559
         2006    454
         2005    450
         2004    433
         2003    422
         Name: yr_built, dtype: int64
```

```
In [137… data1['price'].hist(figsize=(15,8))

Out[137… <matplotlib.axes._subplots.AxesSubplot at 0xff3d170>
```




```
In [138… x = data1.drop(columns=['price'])
         y = data1['price']
```

```
In [139… X_train,X_test,Y_train,Y_test =train_test_split(x,y,random_state=0,train_size=0.80)
```

```
In [233… X_train.shape,X_test.shape

Out[233… ((17290, 17), (4323, 17))
```

```
In [234… linear= LinearRegression()
```

```
In [235… sel= SelectFromModel(LinearRegression())
         sel.fit(X_train,Y_train)

Out[235… SelectFromModel(estimator=LinearRegression(copy_X=True, fit_intercept=True,
                                               n_jobs=None, normalize=False),
                max_features=None, norm_order=1, prefit=False, threshold=None)
```

```
In [236… sel.estimator_.coef_

Out[236… array([-3.31909409e+04,  4.03002875e+04,  1.11182027e+02,  1.71960381e-01,
                 1.80909018e+03,  6.09402810e+05,  4.94058679e+04,  3.07240636e+04,
                 9.47870908e+04,  7.27869231e+01,  3.83951041e+01, -2.45235695e+03,
                 2.16772868e+01,  5.63493777e+05, -1.26112686e+05,  2.97732884e+01,
                -4.76002806e-01])
```

```
In [237… X_train_sel  = sel.transform(X_train)
         X_test_sel = sel.transform(X_test)
```

```
In [238… from sklearn.metrics import r2_score
         from sklearn.tree import DecisionTreeRegressor
```

```
In [239… model = LinearRegression()
         model.fit(X_train_sel,Y_train)
         Y_pred = model.predict(X_test_sel)
         print('R2 Score :',r2_score(Y_test,Y_pred))

R2 Score : 0.5554708605617198
```

```
In [240… corr =data1.corr()['price']
         corr

Out[240… price            1.000000
         bedrooms         0.308350
         bathrooms        0.525138
         sqft_living      0.702035
         sqft_lot         0.089661
         floors           0.256794
         waterfront       0.266369
         view             0.397293
         condition        0.036362
         grade            0.667434
         sqft_above       0.605567
         sqft_basement    0.323816
         yr_built         0.054012
         yr_renovated     0.126434
         lat              0.307003
         long             0.021626
         sqft_living15    0.585379
         sqft_lot15       0.082447
         Name: price, dtype: float64
```

```
In [241… good_corr = corr[corr>0.3]
         len(good_corr)

Out[241… 10
```

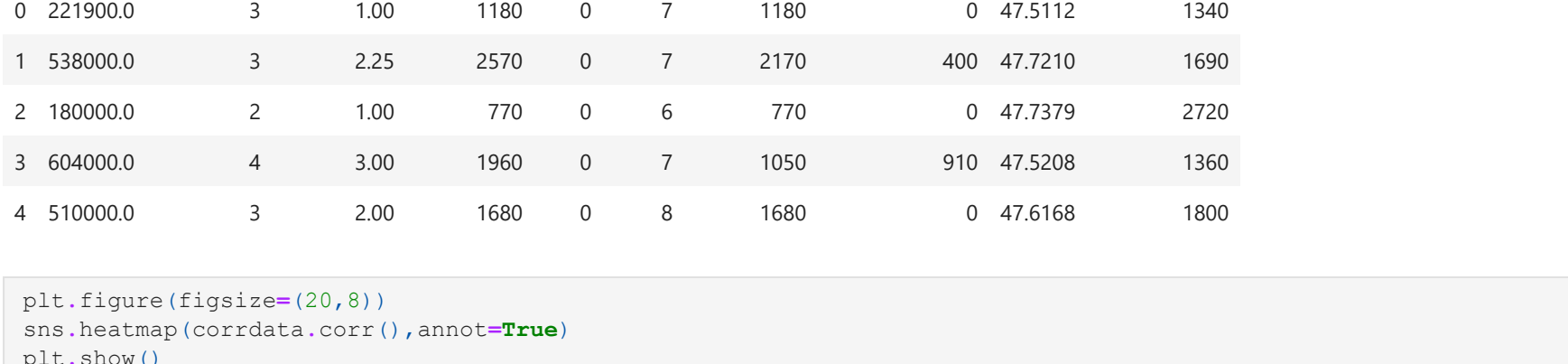
```
In [242… corrdata = data1[good_corr.index]
```

```
In [243… corrdata.head()
```

Out[243…

	price	bedrooms	bathrooms	sqft_living	view	grade	sqft_above	sqft_basement	lat	sqft_living15
0	221900.0	3	1.00	1180	0	7	1180	0	47.5112	1340
1	538000.0	3	2.25	2570	0	7	2170	400	47.7210	1690
2	180000.0	2	1.00	770	0	6	770	0	47.7379	2720
3	604000.0	4	3.00	1960	0	7	1050	910	47.5208	1360
4	510000.0	3	2.00	1680	0	8	1680	0	47.6168	1800

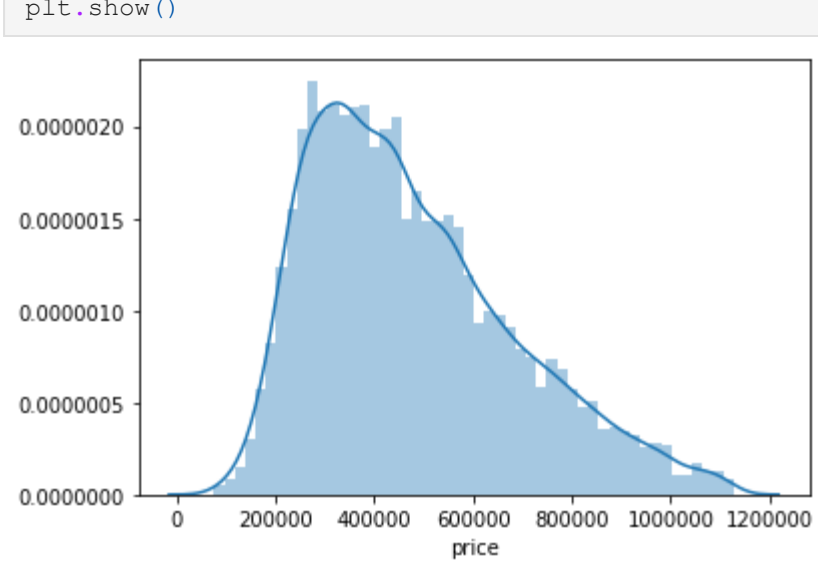
```
In [244… plt.figure(figsize=(20,8))
         sns.heatmap(corrdata.corr(),annot=True)
         plt.show()
```




```
In [245… q1= corrdata['price'].quantile(0.25)
         q3 = corrdata['price'].quantile(.75)

         IQR = q3-q1
         IQR

Out[245… 323050.0
```

```
In [246… dataiqr = corrdata[~((corrdata['price']<q1-1.5*IQR)|(corrdata['price']>q3+1.5*IQR))]
```

```
In [247… sns.distplot(dataiqr['price'])
         plt.show()
```




```
In [248… dataiqr['price'].kurt()

Out[248… -0.0006910700414262116
```

```
In [249… dataiqr['price'].skew()

Out[249… 0.7501833824182833
```

```
In [250… x =dataiqr.drop(columns=['price'])
         y = dataiqr['price']
```

```
In [251… x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=42,train_size=0.80)
```

```
In [252… x_train.shape,y_train.shape

Out[252… ((16373, 9), (16373,))
```

```
In [253… model = LinearRegression()
         model.fit(x_train,y_train)
         y_pred = model.predict(x_test)
         print('R2 score :',r2_score(y_test,y_pred))

R2 score : 0.6288738778109145
```

Here the below graph showing the coeficients are getting zero in the case of Lasso Regression wherease in the case of Ridge regression we will not get exactly zero but we will values in the lower values to some values range and from there we can consider which feature is most important for us.