Understanding the Behavior of Linear Models with SVM and Logistic regression

Chapter 1:

Introduction

1.1 Problem statement

Observing how the behavior of linear models work in case of an imbalanced dataset and implementing the hyper plane on the data and observe how it works for SVM and Logistic Regression.

Checking the feature importance of the numerical feature of the data and comparing it with SVM and Logistic Regression using SGD classifier.

Understanding the impact of outliers by visualizing the best fit linear regression line for different scenarios.

Finding the correlation between the features and find the best model hyper parameter using SVM and Logistic Regression.

1.2 Data

The dataset used is Annual Retail Trade survey which provides the national estimates of total annual sales, operating expenses, and inventories held outside United States.

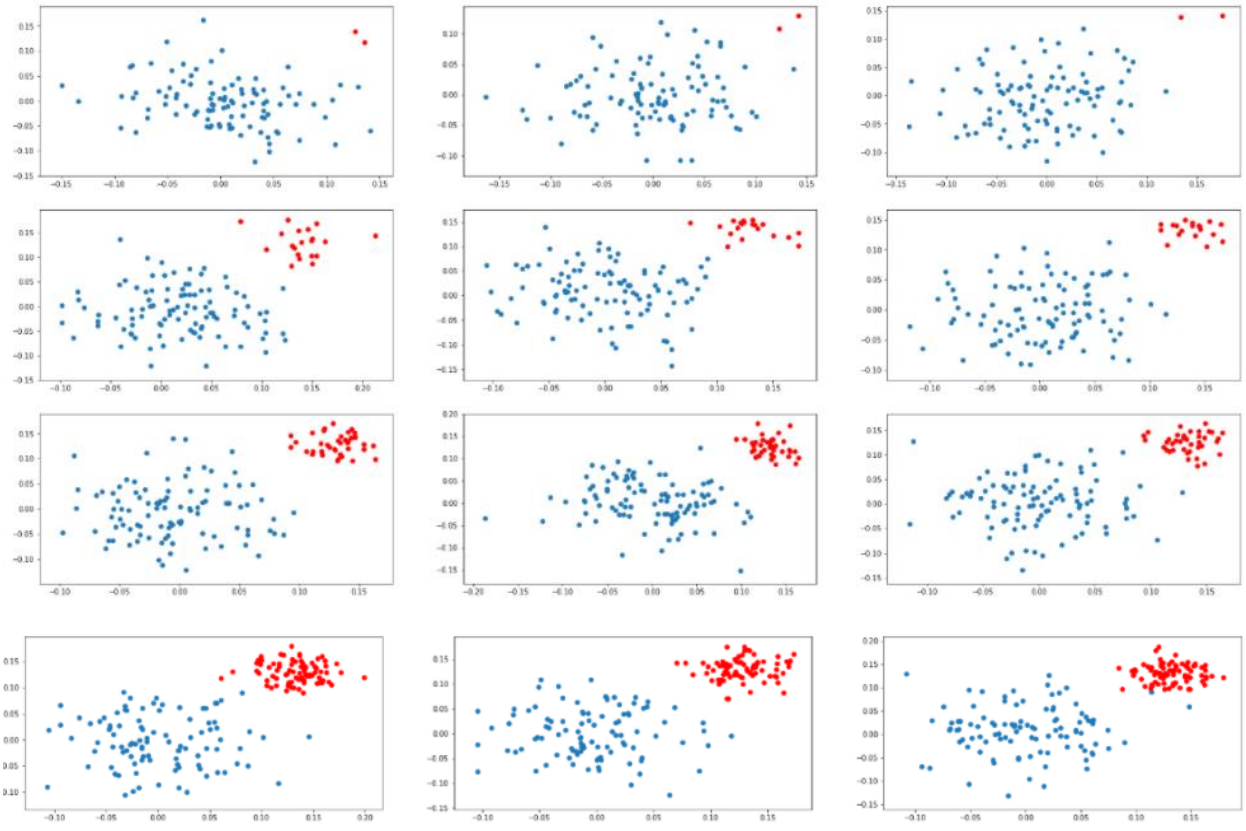
Limited set of data points are considered to implement the Stochastic Gradient Classifier with log loss and hinge loss.

Chapter 2:

Methodology

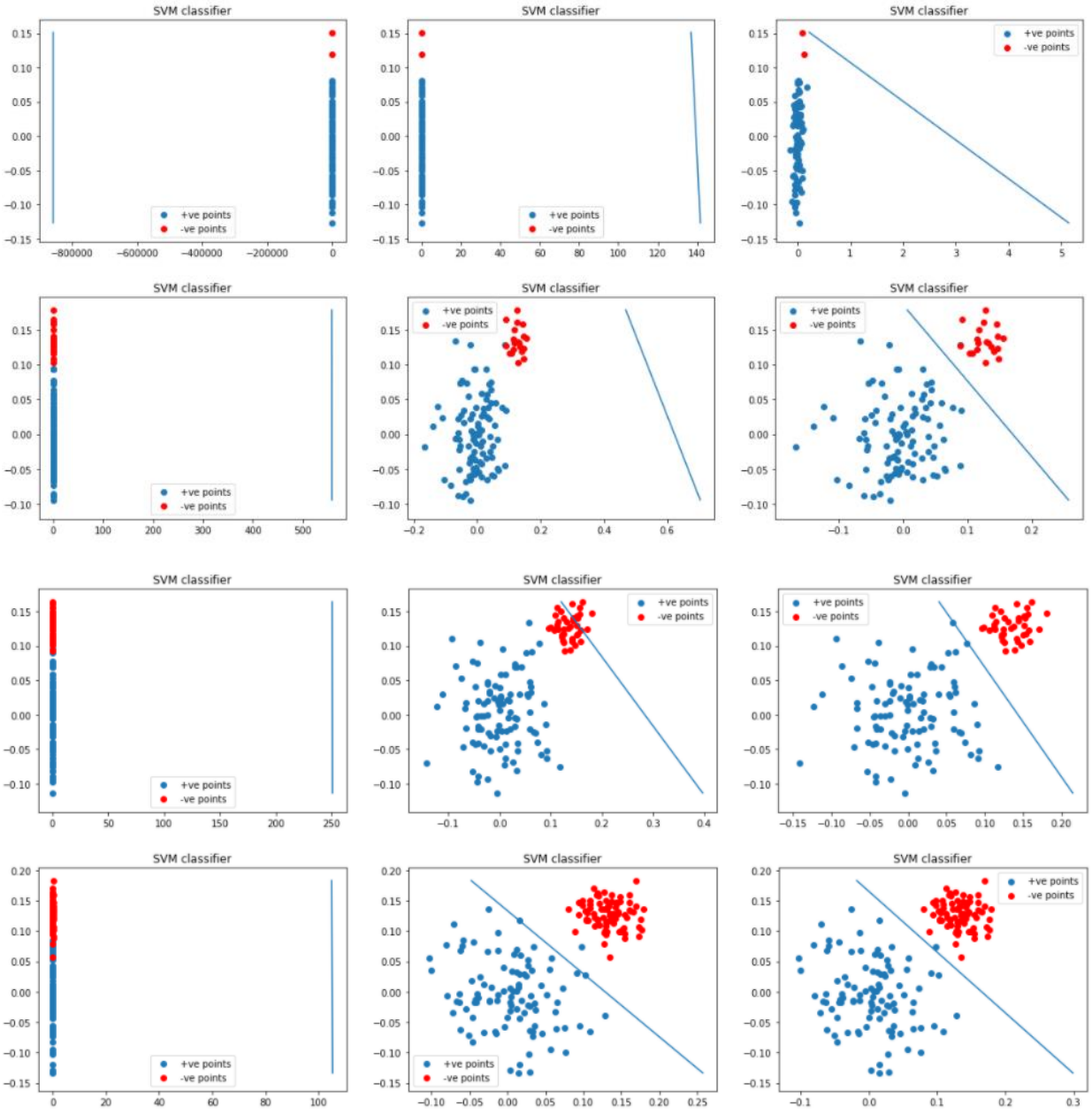
2.1 Behavior of Linear Models

We need to create random data from the main dataset which are linearly separable with ratios between positive and negative as 100:2, 100:20, 100:40, 100:80. Then we need to generate a grid of plots like mentioned below



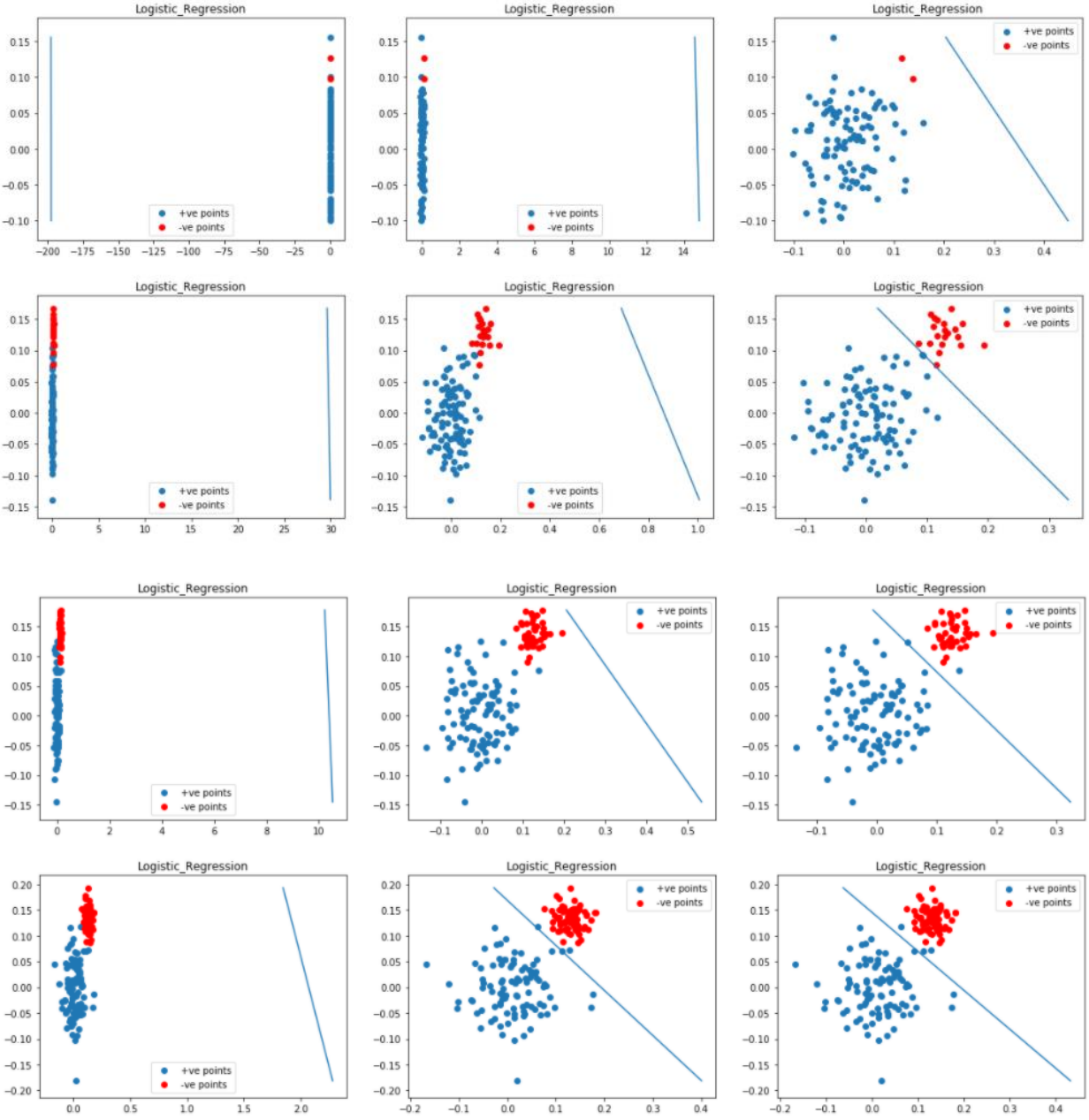
After generating the grid of plots based on the positive and negative ratios we need to draw the hyper plane with different regularizations on both SVM and Logistic Regression and understand the impact of hyper plane.

Support Vector Machine



1. The decision surface do not impact the data, when the parameter value c is low, it can be seen that the plane is strictly towards the right and this leads to under fitting.
2. When the parameter value c is high, it can be observed that the decision surface is fitted properly through the data and able to identify the hyper plane.

Logistic Regression



1. When comparing from the above plots, the performance of Logistic Regression is better than SVM as hyper plane fits to the data when the parameter value c is more.
2. Just like the above, more the value of regularization parameter, better the performance and position of hyper plane.
3. When the parameter value is less, the hyper plane in Logistic Regression tends to bend towards right direction than the
   hyper plane in SVM.

2.2 Feature coefficients

We will observe how the linear models work in case of data with different variance and check the feature importance of the data features by applying Logistic Regression and SVM by SGD classifier with log and hinge loss.

After loading the data, we need to standardize the features and also find the pairwise correlation of all the columns in the data frame.

```
data.head()
```

	f1	f2	f3	y
0	-195.871045	-14843.084171	5.532140	1.0
1	-1217.183964	-4068.124621	4.416082	1.0
2	9.138451	4413.412028	0.425317	0.0
3	363.824242	15474.760647	1.094119	0.0
4	-768.812047	-7963.932192	1.870536	0.0

```
data.corr()['y']
```

```
f1    0.067172
f2   -0.017944
f3    0.839060
y     1.000000
Name: y, dtype: float64
```

```
data.std()
```

```
f1      488.195035
f2    10403.417325
f3        2.926662
y         0.501255
dtype: float64
```

```
standard = StandardScaler()
standard_X = standard.fit_transform(X)
```

Logistic Regression (SGD classifier with log loss)

```
from sklearn.linear_model import SGDClassifier
clf = SGDClassifier(loss='log')
clf.fit(X,Y)
```

```
SGDClassifier(alpha=0.0001, average=False, class_weight=None,
       early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
       l1_ratio=0.15, learning_rate='optimal', loss='log', max_iter=None,
       n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
       power_t=0.5, random_state=None, shuffle=True, tol=None,
       validation_fraction=0.1, verbose=0, warm_start=False)
```

```
clf.coef_, clf.intercept_
```

```
(array([[ 30260.48023888, -10179.34638405,   5663.02223537]]),
 array([-19.63210151]))
```

1. As it can be observed that the features are in both negative and positive values, it implies that negative coefficient of the corresponding feature pushes the classification more towards the negative side and vice versa for positive coefficient.
2. In this model, f1 feature has more positive value than f3 feature. So, "f1" is more important feature and f2 feature is negative which indicates that it pushes the classification towards negative class.

SVM (SGD classifier with hinge loss)

```
clf = SGDClassifier(loss='hinge')
clf.fit(X,Y)
```

```
SGDClassifier(alpha=0.0001, average=False, class_weight=None,
       early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
       l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=None,
       n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
       power_t=0.5, random_state=None, shuffle=True, tol=None,
       validation_fraction=0.1, verbose=0, warm_start=False)
```

```
clf.coef_, clf.intercept_
```

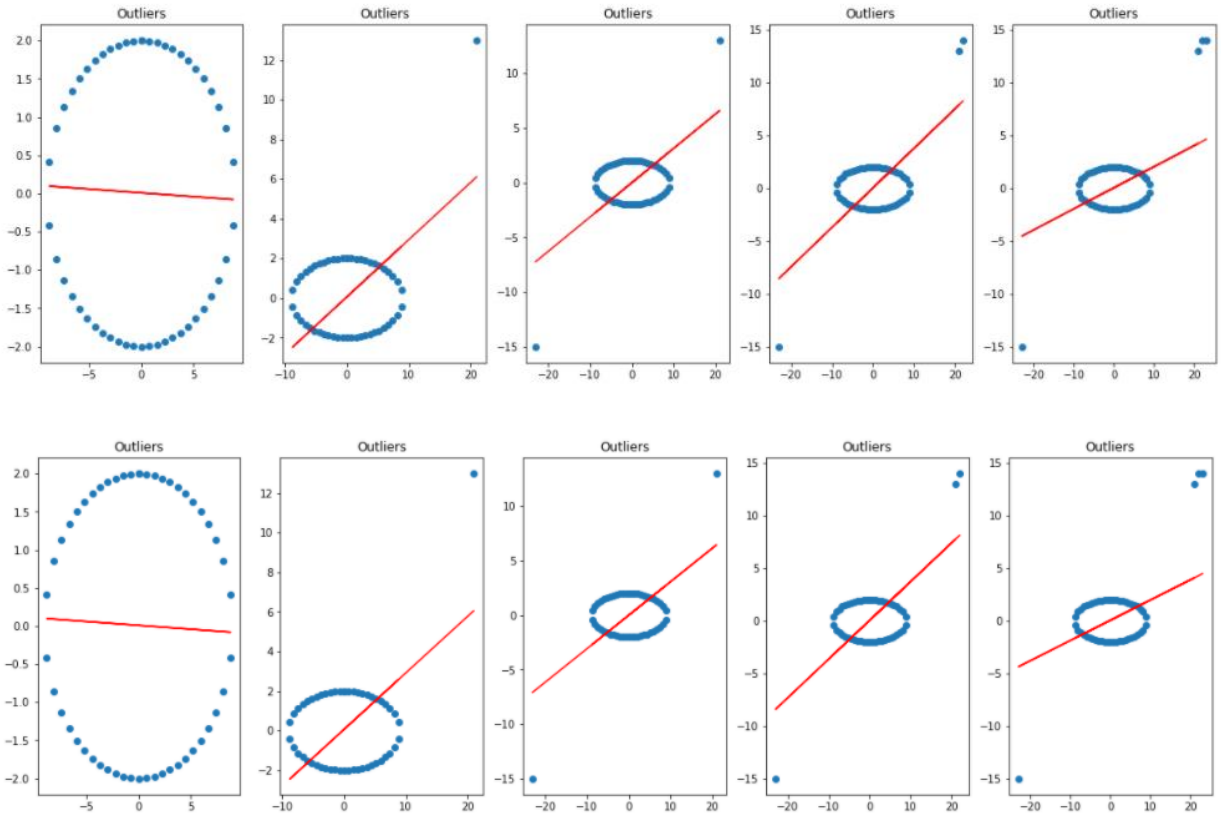
```
(array([[56434.5690072 , 44673.44857989,  6549.6929516 ]]),
 array([81.44233166]))
```

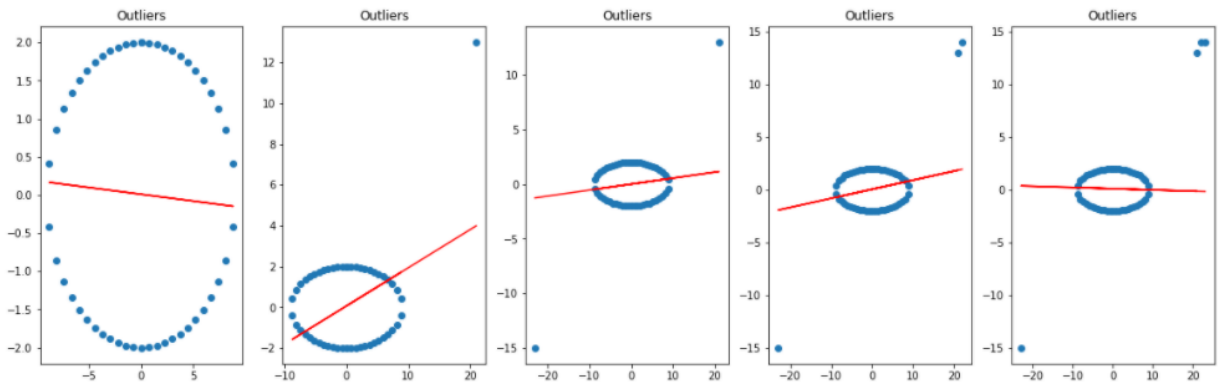
1. Here are the features belongs to positive class, it means all the coefficients pushes the features classification towards positive class.
2. When compared the weight vectors with their corresponding features, feature "f1" is more than f2 and f3, it means "f1" is more important than "f2" and "f3".

2.3 Regression outlier effect

Visualizing the best fit linear regression line for different scenarios and knowing how regularization helps in getting rid of the outliers, considering list of data outliers and plot the hyper plane along with the data points.

```
a_list = [0.0001,1,100]
outlier = [(0,2),(21, 13), (-23, -15), (22,14), (23, 14)]
```





1. when there are no outliers, hyper plane fits perfectly through the data
2. As the number of outliers keeps on increasing from one plot to another, we can observe that the hyper plane tends to deviate towards the outliers direction.
3. So, when the outliers keeps on increasing it affects the direction of the hyper plane, when the alpha is more it can control the impact of outliers up to some extent as it can be observed that the direction of hyper plane is not that much impacted when the value of alpha is more.

2.4 Collinear features and their effect

Finding the best model with their parameters and getting their weights with the coefficients

Logistic Regression

```
lr = SGDClassifier(loss='log')
lr.fit(X,Y)
```

```
SGDClassifier(alpha=0.0001, average=False, class_weight=None,
       early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
       l1_ratio=0.15, learning_rate='optimal', loss='log', max_iter=None,
       n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
       power_t=0.5, random_state=None, shuffle=True, tol=None,
       validation_fraction=0.1, verbose=0, warm_start=False)
```

```
param = {'alpha': np.logspace(0.001,0.01,1,10)}
clf = GridSearchCV(lr,param)
clf.fit(X,Y)
```

```
GridSearchCV(cv='warn', error_score='raise-deprecating',
       estimator=SGDClassifier(alpha=0.0001, average=False, class_weight=None,
       early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
       l1_ratio=0.15, learning_rate='optimal', loss='log', max_iter=None,
       n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
       power_t=0.5, random_state=None, shuffle=True, tol=None,
       validation_fraction=0.1, verbose=0, warm_start=False),
       fit_params=None, iid='warn', n_jobs=None,
       param_grid={'alpha': array([1.00231])}, pre_dispatch='2*n_jobs',
       refit=True, return_train_score='warn', scoring=None, verbose=0)
```

```
best_model = clf.best_params_
print(best_model)
```

```
{'alpha': 1.0023052380778996}
```

Getting the weights with the original data

```
best_model = SGDClassifier(alpha=1.0023052380778996, loss='log')
best_model.fit(X,Y)
```

```
SGDClassifier(alpha=1.0023052380778996, average=False, class_weight=None,
       early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
       l1_ratio=0.15, learning_rate='optimal', loss='log', max_iter=None,
       n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
       power_t=0.5, random_state=None, shuffle=True, tol=None,
       validation_fraction=0.1, verbose=0, warm_start=False)
```

```
from sklearn.metrics import accuracy_score
pred = best_model.predict(X)
best_model_accuracy = accuracy_score(Y, pred)
print(best_model_accuracy)
```

```
1.0
```

```
weights = best_model.coef_
print(weights)
```

```
[[ 0.17102688 -0.18605085  0.25964235  0.16717248 -0.18605085  0.18180396
   0.1513352 ]]
```

SVM

```
svm = SGDClassifier(loss='hinge')
svm.fit(X,Y)
```

```
SGDClassifier(alpha=0.0001, average=False, class_weight=None,
       early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
       l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=None,
       n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
       power_t=0.5, random_state=None, shuffle=True, tol=None,
       validation_fraction=0.1, verbose=0, warm_start=False)
```

```
param_svm = {'alpha': np.logspace(0.001,0.01,1,10)}
clf_svm = GridSearchCV(svm,param_svm)
clf_svm.fit(X,Y)
```

```
GridSearchCV(cv='warn', error_score='raise-deprecating',
       estimator=SGDClassifier(alpha=0.0001, average=False, class_weight=None,
       early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
       l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=None,
       n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
       power_t=0.5, random_state=None, shuffle=True, tol=None,
       validation_fraction=0.1, verbose=0, warm_start=False),
       fit_params=None, iid='warn', n_jobs=None,
       param_grid={'alpha': array([1.00231])}, pre_dispatch='2*n_jobs',
       refit=True, return_train_score='warn', scoring=None, verbose=0)
```

```
best_model_svm = clf_svm.best_params_
print(best_model_svm)
```

```
{'alpha': 1.0023052380778996}
```

Getting the weights with original data

```
best_model_svm = SGDClassifier(alpha=1.0023052380778996, loss='hinge')
best_model_svm.fit(X,Y)
```

```
SGDClassifier(alpha=1.0023052380778996, average=False, class_weight=None,
       early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
       l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=None,
       n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
       power_t=0.5, random_state=None, shuffle=True, tol=None,
       validation_fraction=0.1, verbose=0, warm_start=False)
```

```
from sklearn.metrics import accuracy_score
pred = best_model_svm.predict(X)
best_model_accuracy_svm = accuracy_score(Y, pred)
print(best_model_accuracy_svm)
```

```
1.0
```

```
weights_svm = best_model_svm.coef_
print(weights_svm)
```

```
[[ 0.1607747  -0.22228757  0.35113943  0.1478339  -0.22228757  0.1754457
   0.14108711]]
```