# Predict Winner or not

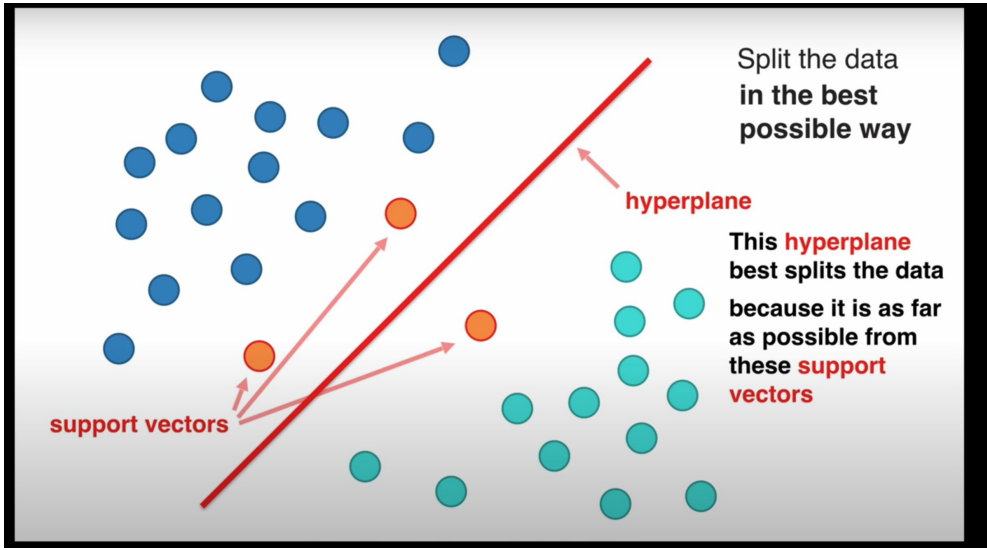
Using Support Vector Machine

# Today’s Discussion

- Define SVM
- Data Description
- We will use SVM to predict if Division 1 winner or not.
- Comparison

# What is SVM

Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges.



# DATA

This is the South African Lottery results from year 2000 when it started to 2015. I was interested in predicting whether there will be winners or not given the following publicly available information prior to betting:

1. Prize Payable
2. Rollover
3. Rollover Count
4. Next Estimated Jackpot

# Loading Data

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import pandas_profiling as pp
%matplotlib inline
%reload_ext autoreload
%autoreload 2
```

```
#Loading Data

df= pd.read_csv("../input/south-african-powerball-results-lottery/POWERBALL.csv")

#View Datafram
df.head()
```

2]:

| | Draw No | Draw Date | Ball 1 | Ball 2 | Ball 3 | Ball 4 | Ball 5 | Powerball | Div1 | Div2 | ... | Powerbal Ball Set | Gauteng Winners | Western Cape Winners | Northern Cape Winners | Eastern Cape Winners | Mpumalanga Winners | Limpopo Winners |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 577 | 29/05/2015 | 4 | 12 | 17 | 25 | 29 | 5 | 0 | 138020 | ... | PB5 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 576 | 26/05/2015 | 12 | 13 | 20 | 31 | 36 | 19 | 0 | 203470 | ... | PB4 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 575 | 22/05/2015 | 9 | 31 | 32 | 40 | 41 | 10 | 0 | 397954 | ... | PB2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 574 | 19/05/2015 | 3 | 21 | 25 | 26 | 36 | 1 | 0 | 236728 | ... | PB5 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 573 | 15/05/2015 | 8 | 26 | 40 | 44 | 45 | 7 | 0 | 273026 | ... | PB4 | 0 | 0 | 0 | 0 | 0 | 0 |

5 rows × 43 columns

# Data Description

[13]:
```
df.describe()
```

Out[13]:

| | Draw No | Ball 1 | Ball 2 | Ball 3 | Ball 4 | Ball 5 | Powerball | Div1 | Div2 | Div3 | ... | Total Sales | Gauteng Winners | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 577.000000 | 577.000000 | 577.000000 | 577.000000 | 577.000000 | 577.000000 | 577.000000 | 5.770000e+02 | 5.770000e+02 | 577.000000 | ... | 5.770000e+02 | 577.000000 | 57 |
| mean | 289.000000 | 7.828423 | 15.448873 | 22.785095 | 30.882149 | 38.173310 | 10.637782 | 2.875913e+06 | 3.465407e+05 | 20861.507799 | ... | 1.572875e+07 | 0.065858 | |
| std | 166.709828 | 6.155679 | 7.510495 | 8.207313 | 7.795786 | 6.135149 | 5.788413 | 1.029682e+07 | 2.402304e+05 | 9135.985327 | ... | 5.274231e+06 | 0.261862 | |
| min | 1.000000 | 1.000000 | 2.000000 | 4.000000 | 5.000000 | 13.000000 | 1.000000 | 0.000000e+00 | 0.000000e+00 | 6310.000000 | ... | 7.404194e+06 | 0.000000 | |
| 25% | 145.000000 | 3.000000 | 9.000000 | 17.000000 | 26.000000 | 35.000000 | 6.000000 | 0.000000e+00 | 1.818430e+05 | 15761.000000 | ... | 1.295400e+07 | 0.000000 | |
| 50% | 289.000000 | 7.000000 | 15.000000 | 23.000000 | 32.000000 | 40.000000 | 11.000000 | 0.000000e+00 | 2.680130e+05 | 19351.000000 | ... | 1.431372e+07 | 0.000000 | |
| 75% | 433.000000 | 12.000000 | 20.000000 | 28.000000 | 37.000000 | 43.000000 | 16.000000 | 0.000000e+00 | 4.375070e+05 | 23369.000000 | ... | 1.653288e+07 | 0.000000 | |
| max | 577.000000 | 33.000000 | 39.000000 | 43.000000 | 44.000000 | 45.000000 | 20.000000 | 1.020166e+08 | 1.121916e+06 | 97278.000000 | ... | 6.034506e+07 | 2.000000 | |

8 rows × 38 columns

[15]:
```
#Number of rows and columns
df.shape
```

Out[15]
```
(577, 43)
```

+ Code + Markdown

# Null Values

[17]:
```
df.isnull().sum()
```

Out[17]:
```
Draw No                     0
Draw Date                   0
Ball 1                      0
Ball 2                      0
Ball 3                      0
Ball 4                      0
Ball 5                      0
Powerball                   0
Div1                        0
Div2                        0
Div3                        0
Div4                        0
Div5                        0
Div6                        0
Div7                        0
Div8                        0
Div1 No Win                 0
Div2 No Win                 0
Div3 No Win                 0
Div4 No Win                 0
Div5 No Win                 0
Div6 No Win                 0
Div7 No Win                 0
Div8 No Win                 0
Prize Payable               0
Rollover                    0
Rollover Count              0
Next Estimated Jackpot      0
Next Guaranteed Jackpot     0
Total Sales                 0
Draw Machine                0
Ball Set                    0
```

# We will use SVM on tha Data.

```
# Division 1 winner or not
#Columns used to predict are Prize Payable, Rollover, Rollover count and Next Estimated Jackpot

y = df.iloc[:, 16:17].values
X = df.iloc[:, 24:28].values
y1 = np.where(y>=1, 1, 0)
```

**Column used to predict:**

Prize Payable

Rollover

Rollover count

Next Estimated Jackpot

# Apply SVM

[54]:
```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y1, test_size = 0.25, random_state = 0)
```

[55]:
```
# Feature Scaling
#Standardize features by removing the mean and scaling to unit variance

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
# Fitting Kernel SVM to the Training set
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', C = 8, random_state = 0)
classifier.fit(X_train, y_train.ravel())
```

Out[56]
```
SVC(C=8, kernel='linear', random_state=0)
```

# ..cont

[57]:
```
# Predicting the Test set results
y_pred = classifier.predict(X_test)
```

[41]:
```
# Predicting the Test set results
y_pred = classifier.predict(X_test)
y_pred
```

Out[41]
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0])
```

[57]:
```
#Comparing the Test Data to the predicted data. This is a 100% Match.
#Turn both arrays to pandas DataFrames and concantenate
y_test = pd.DataFrame(y_test)
y_pred = pd.DataFrame(y_pred)
result_df = pd.concat([y_test, y_pred], axis = 1, sort = False)
result_df
```

Out[58]:

| | 0 | 0 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 0 |
| 2 | 0 | 0 |
| 3 | 0 | 0 |
| 4 | 0 | 0 |
| ... | ... | ... |
| 140 | 0 | 0 |
| 141 | 0 | 0 |
| 142 | 0 | 0 |
| 143 | 0 | 0 |
| 144 | 0 | 0 |

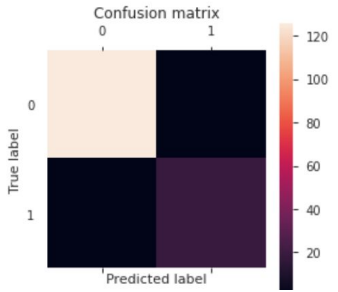
145 rows × 2 columns

# ..cont

[59]:
```
# Making the Confusion Matrix for Visualization of the data
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
score = classifier.score(X_test, y_test)
score
```

Out[59]
```
1.0
```

```
#Plotting the Confusion Matrix
import matplotlib.pyplot as plt
plt.matshow(cm)
plt.title('Confusion matrix')
plt.colorbar()
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```



# Apply k-fold Cross Validation

- k -fold Cross-validation is a resampling **procedure** used to evaluate machine learning models on a limited data sample.
- It ensures that every observation from the original dataset has the chance of appearing in training and test set.

[61]:

```
from sklearn.metrics import accuracy_score
rf_acc_score = accuracy_score(y_test,  y_pred)
print("Accuracy of SVM :",rf_acc_score*100,'\n')
```

```
Accuracy of SVM : 100.0
```

```
# Applying k-Fold Cross Validation to check for mean and Variance
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train.ravel(), cv = 10)
Mean = accuracies.mean()#Mean close to 1
Variance = accuracies.std()#Low Variance
print("Variance and Mean is {0} and {1} ".format(Variance, Mean))
```

```
Variance and Mean is 0.0 and 1.0
```

# Comparing with Logistic Regression

[20]:
```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y1, test_size = 0.25, random_state = 0)
```

[21]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
model = LogisticRegression()
model.fit(X, y)
predicted_classes = model.predict(X)
accuracy = accuracy_score(y.flatten(),predicted_classes)
parameters = model.coef_
accuracy*100
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/utils/validation.py:63: DataConversionWarning: A column-vector y was pass
ed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
```

Out[21]
```
98.44020797227037
```

# SVM vs Logistic Regression

**SVM** tries to finds the “best” margin (distance between the line and the support vectors) that separates the classes and this reduces the risk of error on the data, while **logistic** regression does not, instead it can have different decision boundaries with different weights that are near the optimal point.