Analytics using Python

Learning outcomes

1. You will learn Python , a useful language

2. Use programming for building LDA and SVM models

Great Lakes Institute of Management

A guide to learn python for analytics

P. V. Subramanian

**A workbook on Analytics using Python**

**P. V. SUBRAMANIAN**

**Visiting faculty – Great Lakes Institute of Management**

*Copyright @ 2019 Great Lakes Institute of Management : All contents in this book, including (but not limited to) its contents are proprietary to Great Lakes Institute of Management, Chennai. Any effort at re-using, modifying and circulating both through Digital or Physical means, without the express written permission of the author, will amount to infringement of copyrights, and will be liable for suitable action.*

**Chapter 12. ADVANCED PREDICTIVE MODELLING**

**Table of Contents**

[1. LINEAR DISCRIMINANT ANALYSIS 4](#_Toc18877446)

[1.1. Introduction 4](#_Toc18877449)

[1.2. WHAT LDA does? 4](#_Toc18877450)

[1.3. HOW LDA WORKs? 4](#_Toc18877451)

[1.4. Comparison ANOVA / Regression WITH LDA 5](#_Toc18877452)

[1.5. Comparison between PCA and LDA 5](#_Toc18877453)

[1.6. When to use LDA? 5](#_Toc18877454)

[1.7. Business applications of LDA 6](#_Toc18877455)

[1.8. Objectives of the discriminant analysis 6](#_Toc18877456)

[1.9. Linear Discriminant Model 6](#_Toc18877457)

[1.10. Standardized, Unstandardized and Structure coefficients 7](#_Toc18877458)

[1.11. Calculation of discriminant coef 7](#_Toc18877459)

[1.12. Graphical representation of LDA 8](#_Toc18877460)

[1.13. Statistics associated with Discriminant Analysis 8](#_Toc18877461)

[1.14. Feature importance using LDA 9](#_Toc18877462)

[1.15. LDA EXAMPLE using PYTHON 9](#_Toc18877478)

[1.15.1 Fisher Discriminant Model 17](#_Toc18877495)

[2. SUPPORT VECTOR MACHINES (SVM) 25](#_Toc18877496)

[2.1. Introduction to SVM 25](#_Toc18877498)

[2.1.1 How does SVM work? 25](#_Toc18877501)

[2.1.2 Advantages and disadvantages of SVM 26](#_Toc18877502)

[2.1.3 Applications of SVM 26](#_Toc18877503)

[2.1.4 K Nearest Neighbors (KNN) Vs Support Vector Machines (SVM) 27](#_Toc18877504)

[2.1.5 SVM Classifiers 28](#_Toc18877505)

[2.1.6 SUPPORT VECTOR REGRESSION 59](#_Toc18877506)

# LINEAR DISCRIMINANT ANALYSIS



## Introduction

* Linear discriminant analysis (LDA) is a dimensionality reduction technique used while pre-processing for machine learning applications.
* Goal of LDA is to project a higher-dimensional dataset onto a lower-dimensional space with clear division of classes.
* Ronald r fisher formulated linear discriminant in 1936 (the use of multiple measurements in taxonomic problems).

## WHAT LDA does?

* LDA reduces dimensions similar to PCA.
* LDA searches for a linear combination of variables that best separates two classes
* Reduces overfitting  
  In short, LDA algorithm reduces the dimensions of the data in a way that inter class separation is also taken care of.

## HOW LDA WORKs?

* LDA derives the coefficients of a scoring function for each category from the input data.
* Each function takes as arguments the numeric predictor variables of a class.
* LDA then scales each variable according to its category-specific coefficients and outputs a score.
* The LDA model uses the highest score to allocate a class to a category (prediction).
* These scoring functions are known as discriminant functions.

<https://www.displayr.com/linear-discriminant-analysis-in-r-an-introduction/>

## Comparison ANOVA / Regression WITH LDA

* ANOVA and Regression analysis like LDA also attempt to express one dependent variable as a linear combination of independent variables.
* ANOA and Regression use categorical independent variables and a continuous dependent variable. LDA use categorical dependent variable and continuous independent variables.

## Comparison between PCA and LDA

* LDA approach is very similar to a Principal Component Analysis since both are dimensionality reduction techniques.
* In LDA, in addition to finding the component axes that maximize the variance of data (what PCA does), we find the axes that maximize the separation between multiple classes.

**PCA AND LDA COMPARISON**

|  |  |
| --- | --- |
| **PRINCIPAL COMPONENT ANALYSIS (pCA)** | **lda** |
| PCA is an unsupervised learning algorithm since it ignores the class labels that maximize the variance in a dataset, to find the directions. | 1. LDA is a supervised algorithm since it takes the class label into consideration. 2. It is a way to reduce ‘dimensionality’ while at the same time preserving as much of the class discrimination information as possible. |
| PCA infers some new features, which best describe the type of class from the existing features. It does not select a set of features and discard other features. | 1. LDA helps you find the boundaries around clusters of classes. 2. It projects your data points on a line so that your clusters are as separated as possible, with each cluster having a relative (close) distance to a centroid. |

## When to use LDA?

* Y, dependent variable is categorical variable with more than two levels
* X, a set of independent variables which are numerical and normally distributed

## Business applications of LDA

* Predict solvency or insolvency of a company based on financial ratios
* Identification of type of customers who are likely to cancel the membership or subscription or buy a product or subscribe a magazine.
* Pattern recognition - For example, to distinguish objects, humans, cars, dogs, etc.
* For most of the classification problems, LDA / QDA techniques will work.

## Objectives of the discriminant analysis

* Development of discriminant functions, or linear combinations of the predictor or independent variables, which will best discriminate between categories of the criterion or dependent variable (groups)
* Examination of whether significant differences exist among the groups, in terms of the predictor variables
* Determination of which predictor variables contribute to most of the intergroup differences
* Classification of cases to one of the groups based on the values of the predictor variables

## Linear Discriminant Model

**The LDA model gives linear combinations of the predictor variables as follows:**

* DS = β0 + β1X1 + β2X2 + β3X3 + ... + βk Xk
* Where:
* DS = Discriminant Score
* β 's = Discriminant weight (coefficients)
* X’s = Explanatory (Predictor or independent) variables
* Note:

1) The weights (or coefficients) are estimated so that the groups are separated as clearly as possible on the values of the discriminant functions.

2) LDA constructs an equation which minimizes the possibility of misclassifying cases into their respective classes.

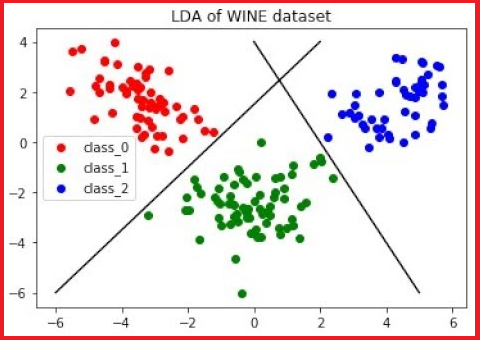
## Standardized, Unstandardized and Structure coefficients

* Discriminant functions are interpreted by means of standardized coefficients and the structure matrix.
* **Unstandardized (raw) coefficients** produced by the analysis when the analysis is performed on unstandardized, original variables.
* **Standardized coefficients** are unit free and scaled to have mean zero and a standard deviation of 1.
* Unstandardized coefficients are less useful for direct comparison when unit of measure for the predictor variables differ. Standardized coefficients represent each independent variables' weight in the discriminant function. The larger the standardized beta coefficient, larger is the respective variable's unique contribution to the discrimination as specified by the respective discriminant function.
* **Structure correlation coefficients** give the correlation between each independent variable and the discriminant score of each function. Using these, we can identify the independent variables which cause the discrimination between dependent variables.
* For more details, please refer: <https://en.wikipedia.org/wiki/Linear_discriminant_analysis>

## Calculation of discriminant coef

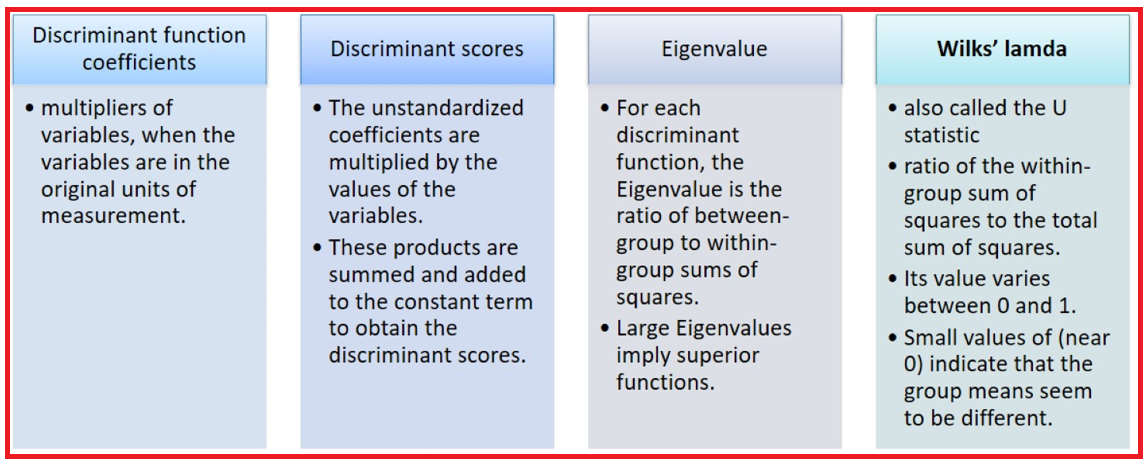
* Maximize the ratio of the variation between the classes and the variation within the classes
* Points belonging to the same class should be close together, while also being far away from the other classes

## Graphical representation of LDA



* Refer: <https://www.youtube.com/watch?v=qiq4Y0ZkAwU>
* You can appreciate how well LDA discriminate the three classes (cultivars)

## Statistics associated with Discriminant Analysis



## Feature importance using LDA

* There are multiple ways to measure feature importance. Here, we will focus on permutation importance.
* Permutation importance is calculated after a model has been fitted.
* Choose a single column in turn for finding the permutation importance.
* Shuffle the values in the chosen column, make predictions using the resulting dataset. Use these predictions and the true target values to calculate how much the loss function suffered from shuffling.
* The performance deterioration measures the importance of the variable you have just shuffled.
* Do the above process for all columns.

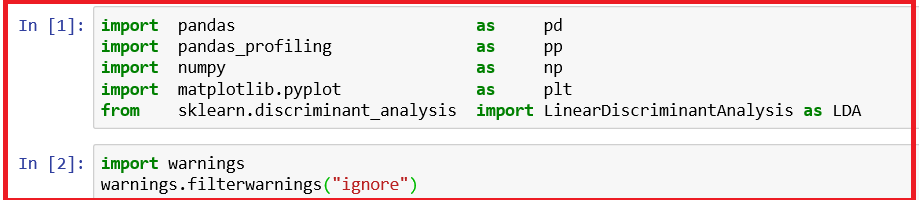


## LDA EXAMPLE using PYTHON

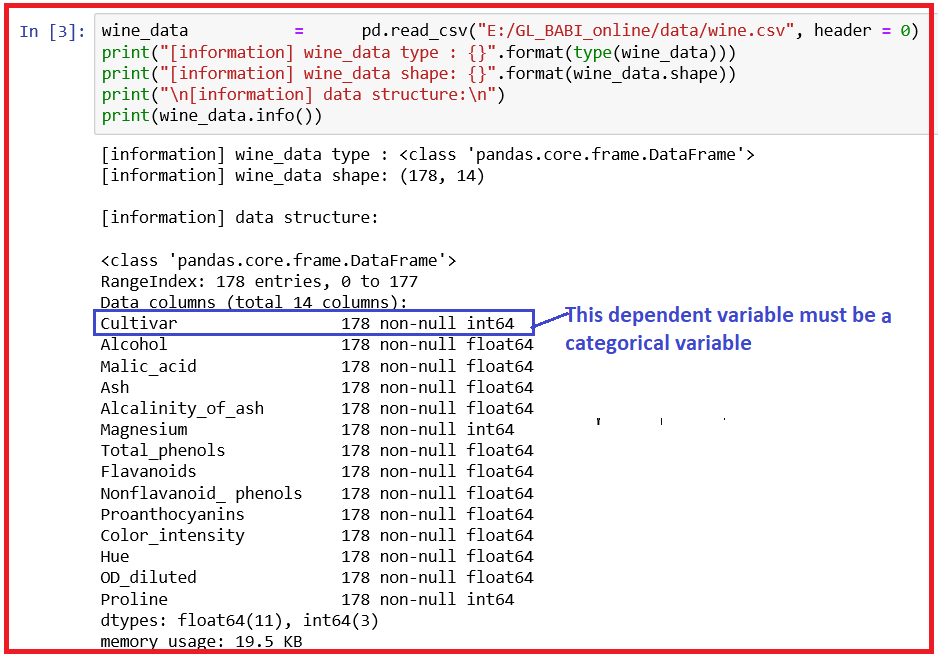
* Use the wine data set containing wine samples from three cultivars having 13 chemical concentrations. Construct a classifier, which determines to which cultivar a specific wine sample belongs.

|  |  |
| --- | --- |
| ***Source: Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.***  **Cultivar is a variety of a plant developed from natural species and maintained under cultivation.** | **Attributes**  ***1. Alcohol***  ***2. Malic acid***  ***3. Ash***  ***4. Alcalinity of ash***  ***5. Magnesium***  ***6. Total phenols***  ***7. Flavanoids***  ***8. Nonflavanoid phenols***  ***9. Proanthocyanins***  ***10. Color intensity***  ***11. Hue***  ***12. OD280/OD315 of diluted wines***  ***13. Proline*** |

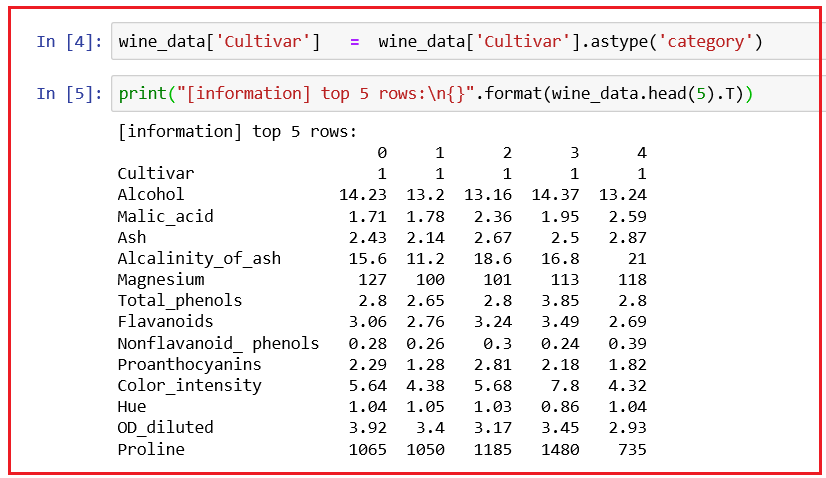
* The goal of LDA is to find combination of the variables that separates the three wine cultivars clearly. The ‘.’ in the formula implies that we use all the remaining variables in the data as covariates.
* import statement allows you to load one or more modules into your Python program, letting you make use of the definitions constructed in those modules. Load pandas for loading and manipulating data, matplotlib.pyplot for visualization and sklearn,discriminant\_analysis to perform LDA in Python.

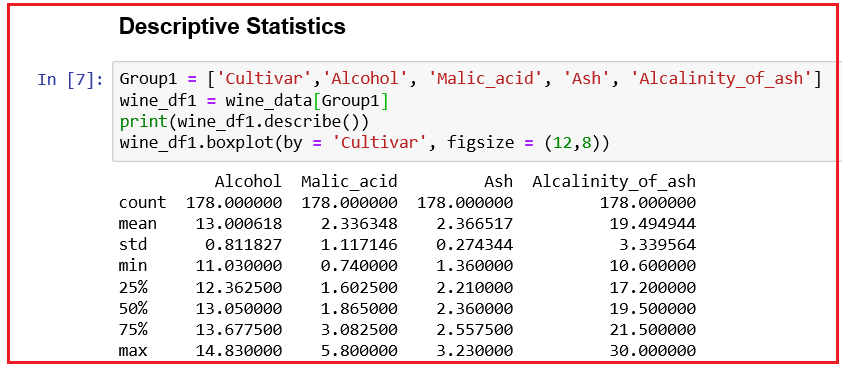


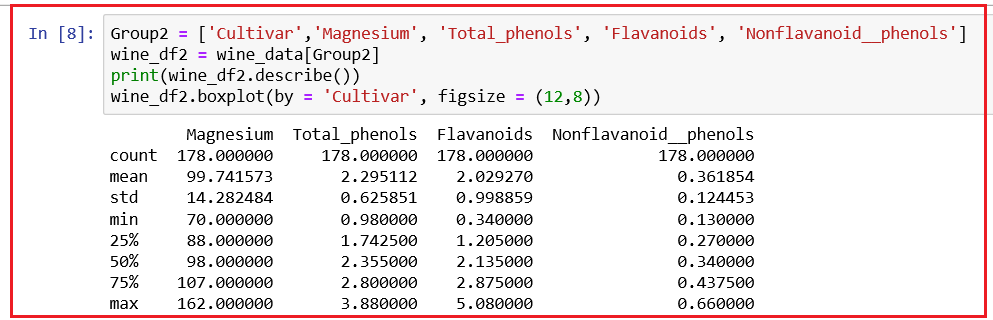
**Read data**

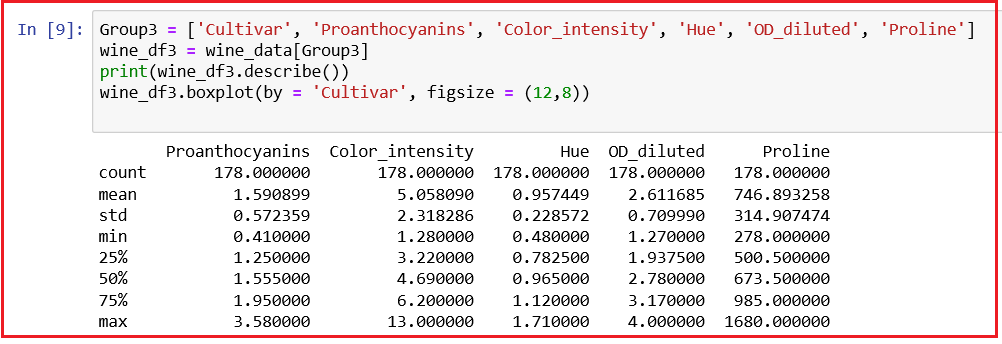


* We have 178 observations and 14 columns including both predictor and response variable.
* We need to convert the target variable from integer to categorical variable,



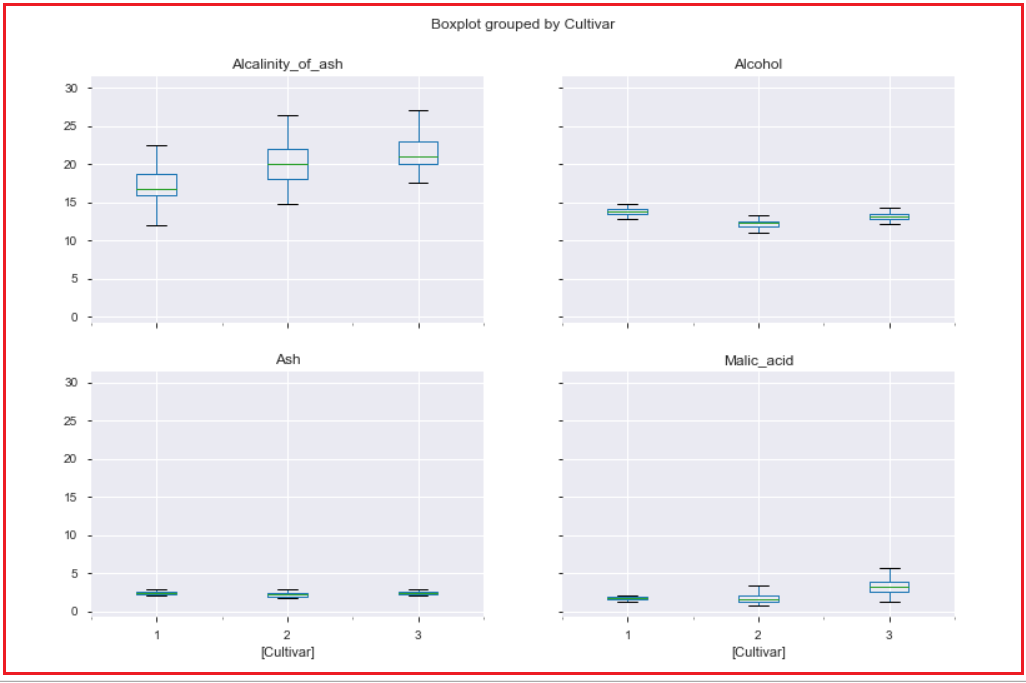
* Let us get descriptive statistics for our wine data.





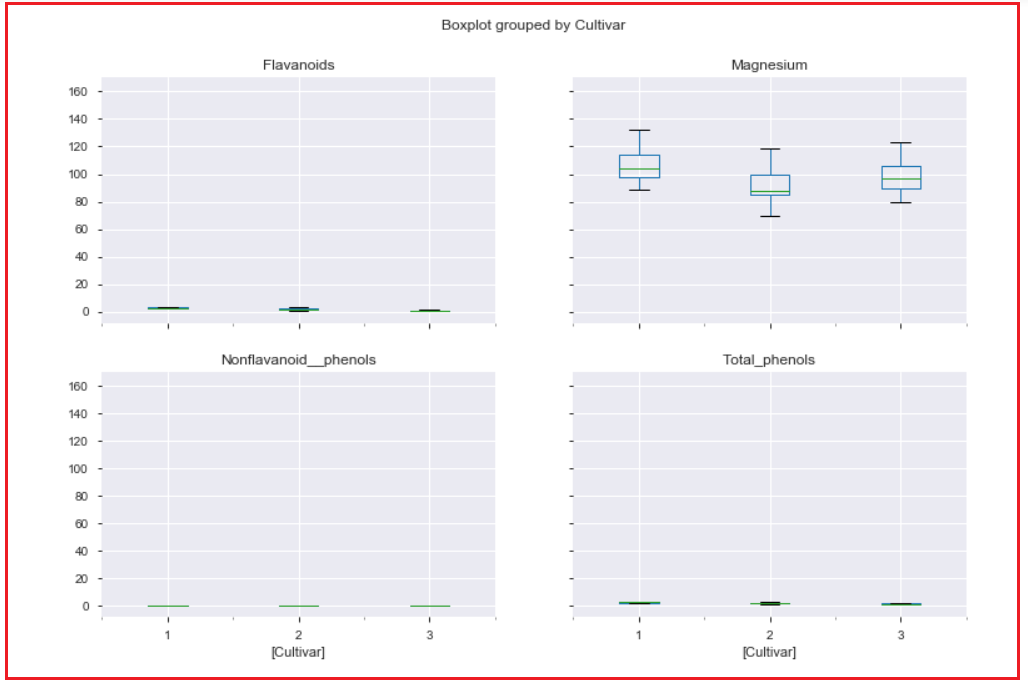
***We observe some substantial difference between mean and median for the variables, Malic\_acid, color\_intensity and Proline.***

**Data Profiling**

* Let us do data profiling to understand the data better

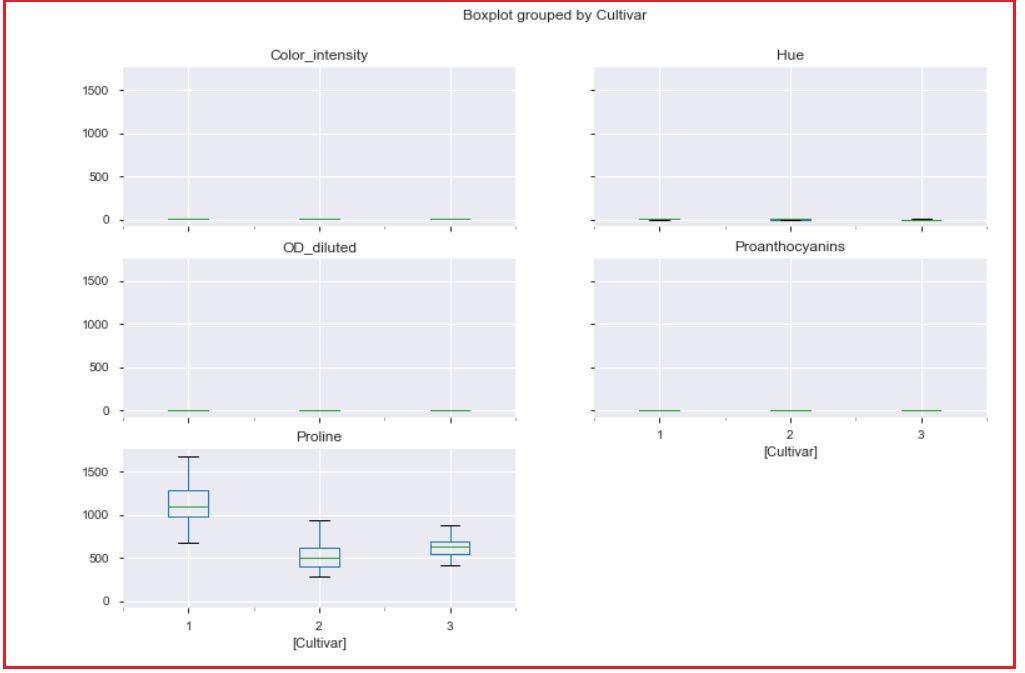
**Interpretation:**

* All the three Cultivars 1, 2 and 3 show substantial differences with regard to Alcalinity\_of\_ash when compared to Alcohol, Ash and Malic\_acid.
* We can draw similar boxplots for all other independent variables.



**Interpretation:**

* All the three Cultivars 1, 2 and 3 show substantial differences with regard to Magnesium when compared to Flavanoids, Nonflavanoid\_phenols and Total\_phenols.



**Interpretation:**

* All the three Cultivars 1, 2 and 3 show substantial differences with regard to Proline when compared to Proanthocyanins, Color\_intensity, Hue, OD\_diluted.
* At a glance, Proline appears to be a differentiator of the Cultivator Groups.

**LDA in scikit learn**

* Linear Discriminant Analysis is a classifier with a linear decision boundary, generated by fitting class conditional densities and using Bayes' rule.
* The model fits a Gaussian density to each class, assuming that all classes share the same covariance matrix.
* The fitted model can also be used to reduce the dimensionality of the input by projecting it to the most discriminative directions.

**Main parameters to the function, LinearDiscriminantAnalysis**

1. **Shrinkage:** It is a tool to improve estimation of covariance matrices in situations where the number of training samples is small compared to the number of features. Currently, shrinkage only works when setting the solver parameter to lsqr or eigen.

**Possible values for shrinkage:**

1. **None -** No shrinkage (default)
2. **auto -** Automatic shrinkage using the Ledoit-Wolf lemma
3. **float -** value between 0 and 1, a fixed shrinkage parameter

Refer https://scikit-learn.org/stable/modules/lda\_qda.html

1. **solver: It is the solver to use**

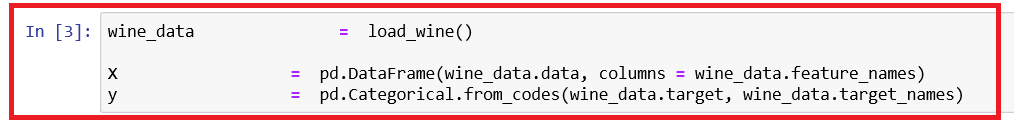
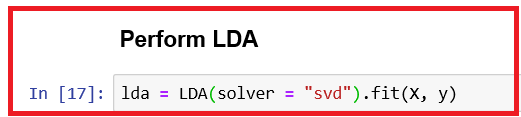
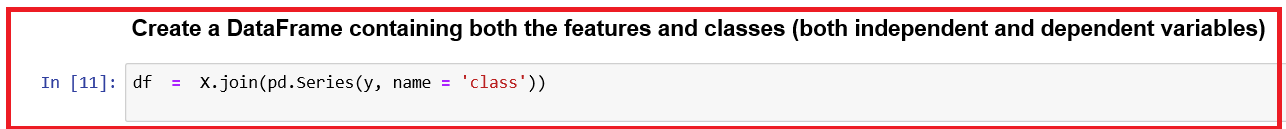
**Possible values:**

1. **svd** -- Singular Value Decomposition. This is the default value. This does not compute the covariance matrix; hence this solver is recommended for data with a large number of features
2. **lsqr** -- Least Squares Solution is an efficient algorithm that only works for classification and it supports shrinkage.
3. **eigen** -- Eigen Value decomposition is based on the optimization of the between class scatter to within class scatter ratio. It can be used for both classification and transform, and it supports shrinkage. However, the eigen solver needs to compute the covariance matrix, so it might not be suitable for situations with a high number of features.

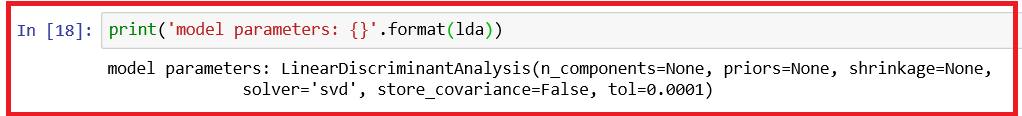
### Fisher Discriminant Model

* We want to separate the wines by cultivar.
* The wines come from three different cultivars, so the number of groups G = 3, and the number of variables is k which is 13
* The maximum number of useful discriminant functions that can separate the wines by cultivar is the minimum of (G−1, k).
* Hence, we can find at most 2 useful discriminant functions to separate the wines by cultivar, using the 13 chemical concentration variables.

**Build Model**

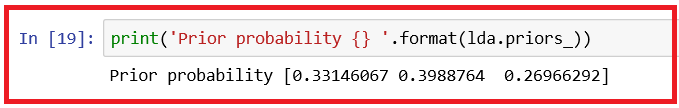
****

**Model parameters**



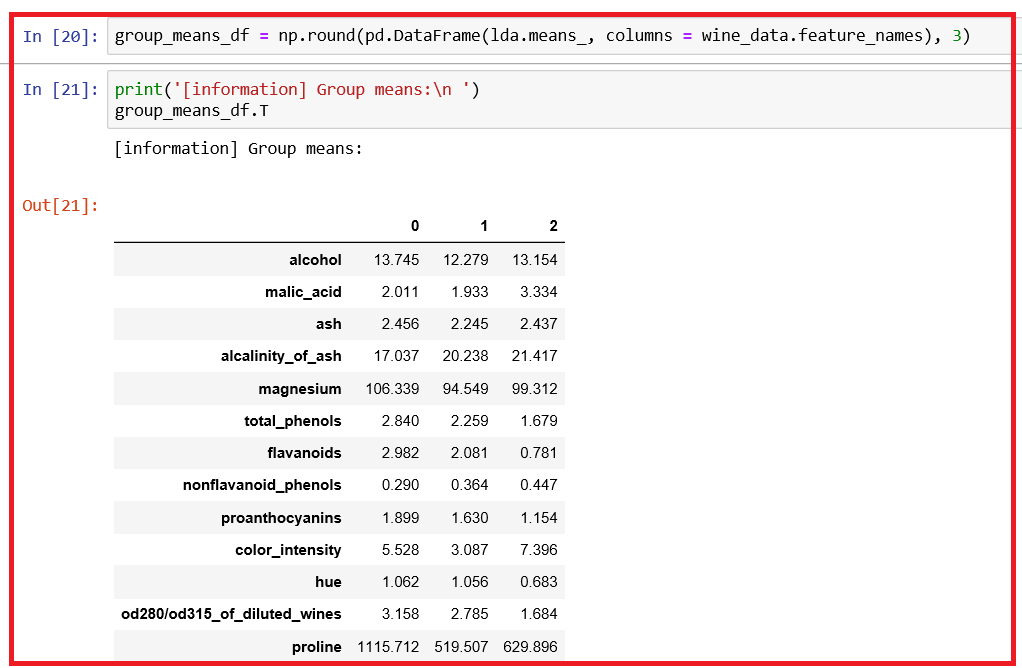
**Prior Probability**

* The prior probability of a class is usually estimated by empirical frequencies of the class in the data set.

****

* We observe that the prior probabilities of three cultivars are not equal.
* Prior probabilities are the ones that already exists in your data. That is around 33% of data corresponds to cultivar 1, nearly 40 % of the data corresponds to cultivar 2 and approximately 27% of the data corresponds to cultivar 3.

**Group means**

****

* In Group means table contains the average of each predictor within each class. These values could suggest that the variables such as Alcohol, Magnesium, Color\_intensity, od280/od315\_of\_diluted\_wines and Proline might have a slightly greater influence on Cultivar 1 than other Cultivars.
* Difference is very large between Cultivar 1 and the rest for these two independent variables.

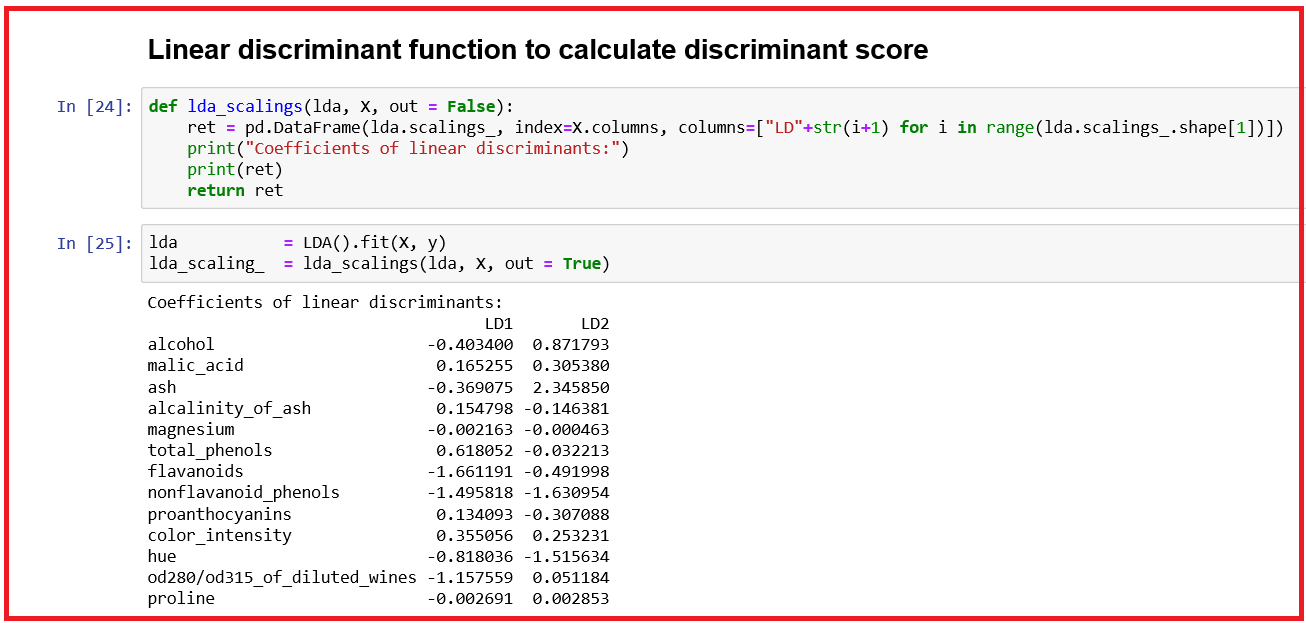
**Proportion of Trace**

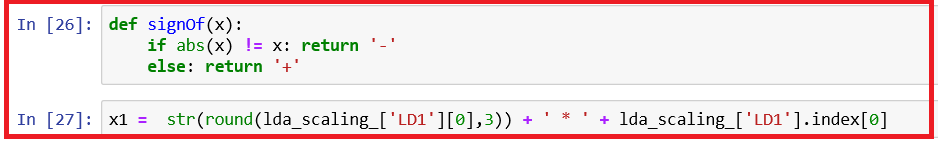
****

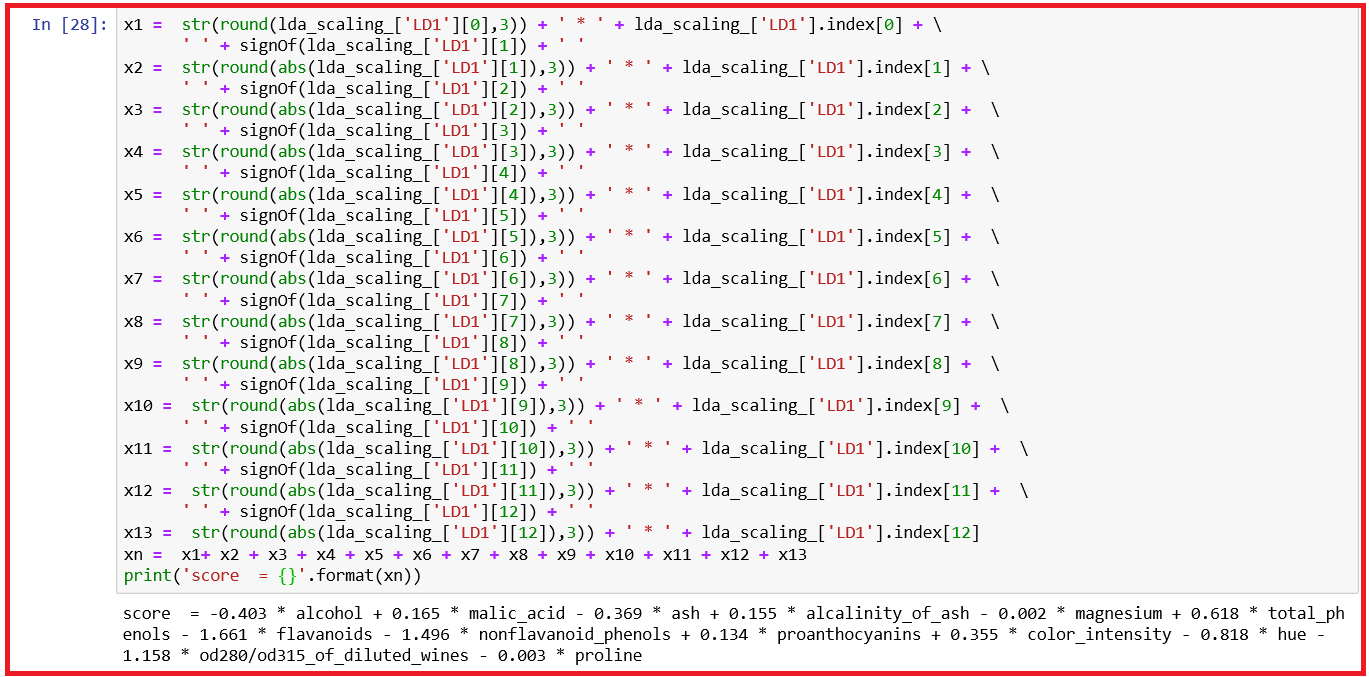
* Since LDl captures 68.75% of the information content (See Proportion of trace above), it should be used for understanding differentiation of predictor variables and LD2 should be discarded as it is weak in terms of capturing information content.

**Coefficients of Linear Discriminants**

* This shows the linear combination of predictor variables that are used to form the LDA decision rule.

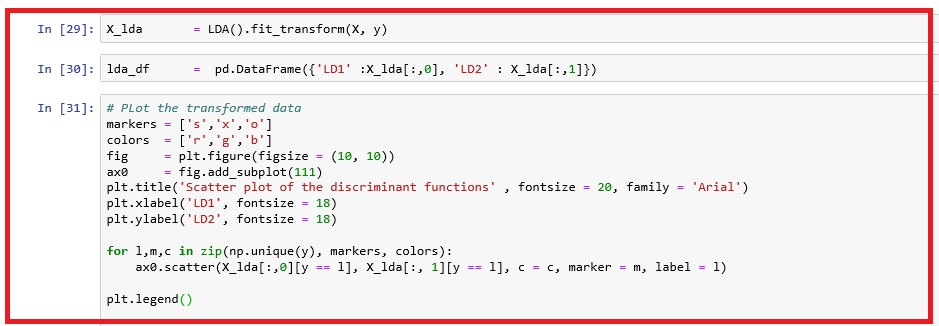
****

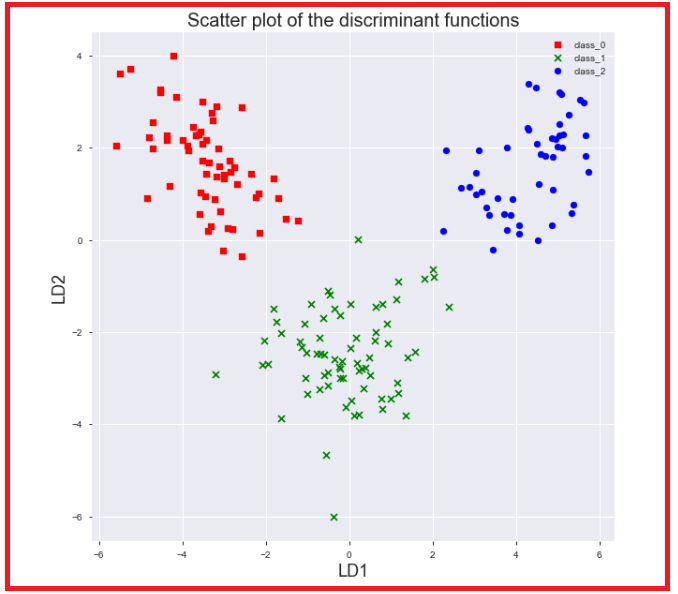
****

****

* From the coefficients of the discriminants, we have shown the first discriminant function, LD1 is a linear combination of the variables. The value for each discriminant function is scaled so that their mean value is zero.
* These loadings are calculated so that the within-group variance of each discriminant function for each group is equal to 1.
* These scalings are stored in scalings of the object returned by LDA().fit(X, y). This element contains an Numpy array, in which first column contains the loadings of the first discriminant function, the second column contains the loadings for the second discriminant function and so on.

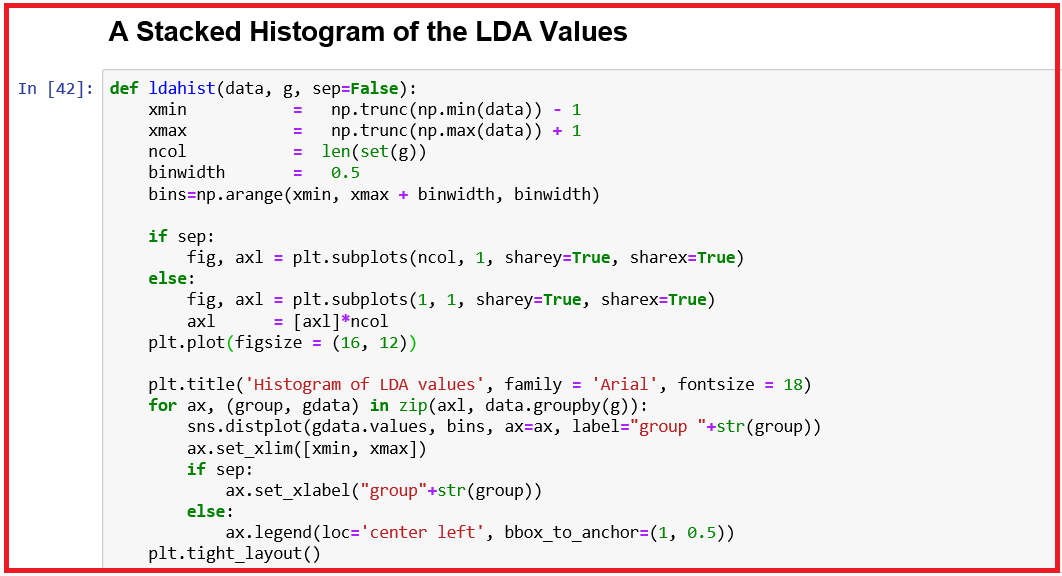
**Plot the standardized discriminant function**

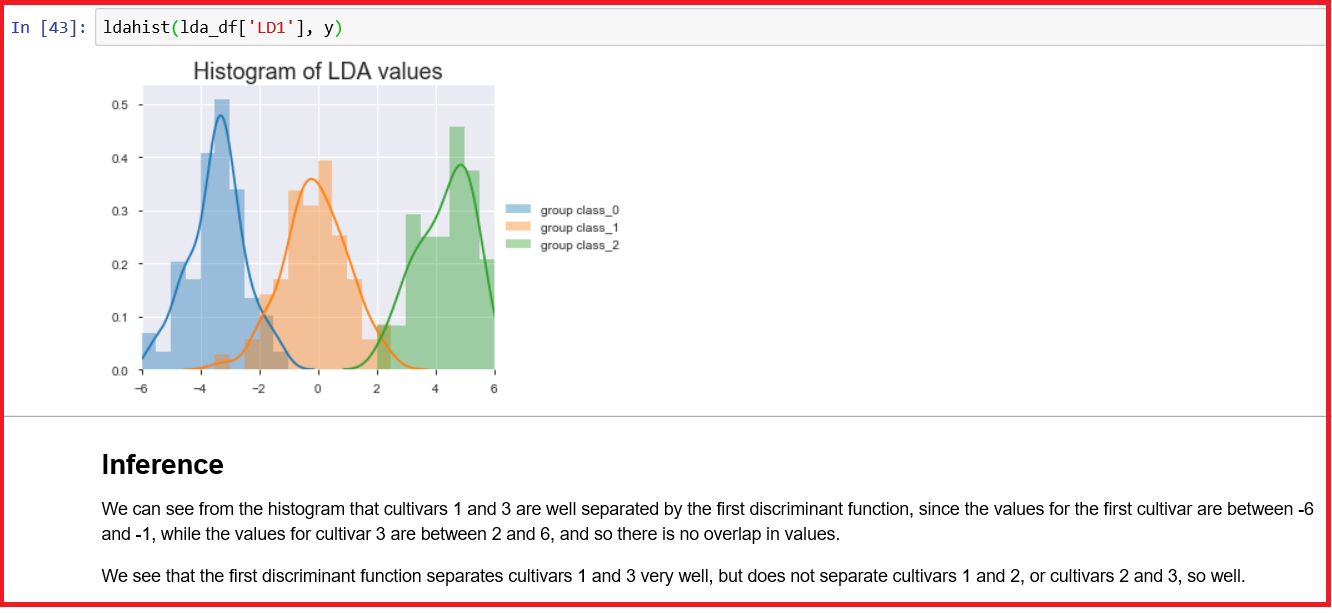




* From the scatterplot of the first two discriminant functions, we can see that the wines from the three cultivars are well separated in the scatterplot.
* The first discriminant function (x-axis) separates cultivars 1 and 3 well, but doesn’t not perfectly separate cultivars 1 and 3, or cultivars 2 and 3.

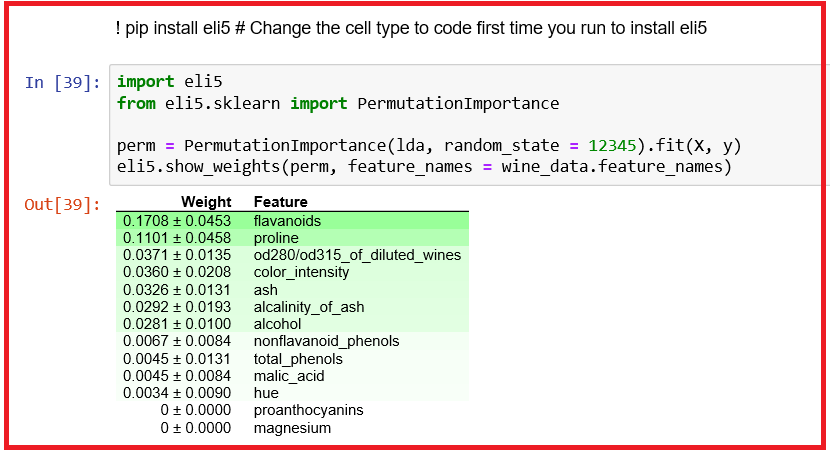
**A Stacked Histogram of the LDA Values**

****

****

**Permutation importance**

* One basic question one would ask of a model is What features have big influence on predictions?
* This concept is feature importance. Here, we will focus on permutation importance.
* Permutation importance is calculated after a model has been fitted.
* Shuffle the values in a single column, make predictions using the resulting dataset. Use these predictions and the true target values to calculate how much the loss function suffered from shuffling.
* The performance deterioration measures the importance of the variable you have just shuffled.
* Do the above process for all columns.



**Inference**

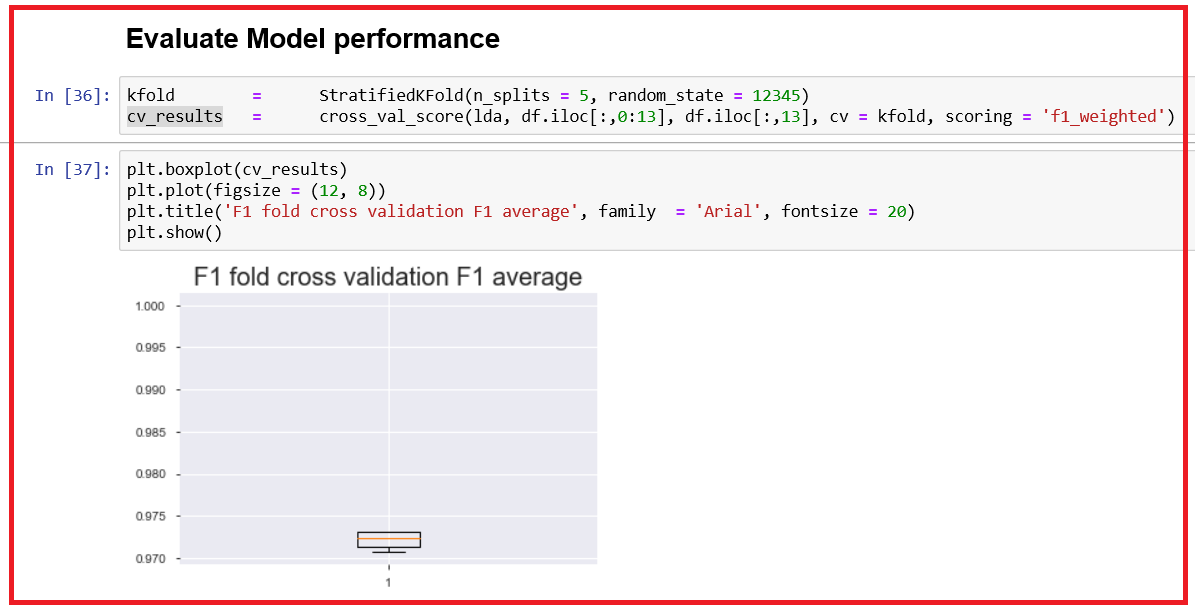
* We observe that flavanoids and proline are important factors in determining the cultivars of wine.
* In the context of the compound structure of wine grapes, flavonoids are complicated phenolic compounds that give wine grapes specific characteristics. Flavonoids are responsible for the color and flavor of grapes, which determines the wine style that the grapes will be best suited for. In brewing, proteins rich in proline combine with polyphenols to produce haze (turbidity)\*. Ref: https://en.wikipedia.org/wiki/Proline

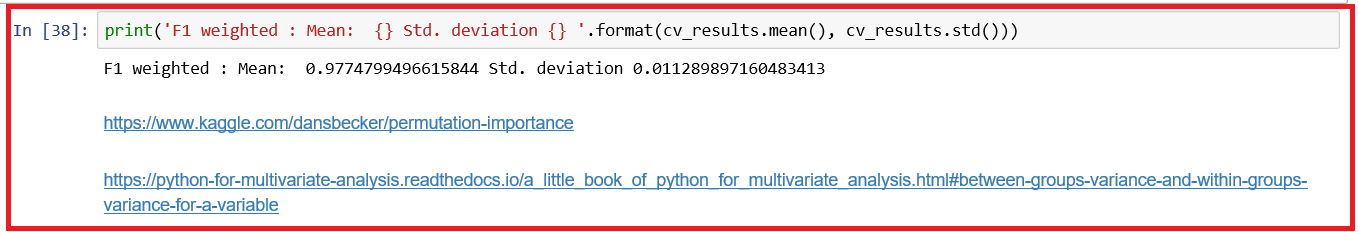
\* K.J. Siebert, "Haze and Foam","Archived copy". Archived from the original on 2010-07-11. Retrieved 2010-07-13. Accessed July 12, 2010.

* Refer: <https://www.kaggle.com/dansbecker/permutation-importance>
* <https://python-for-multivariate-analysis.readthedocs.io/a_little_book_of_python_for_multivariate_analysis.html#between-groups-variance-and-within-groups-variance-for-a-variable>

**Evaluate Model performance**

* We shall use cross-validation instead of splitting the data into train and test data sets.
* By using cross-validation, you verify how accurate your model is on several and different subsets of data.
* This ensures that it generalizes well to the data that you collect in the future. It does improve the accuracy of the model and reduces overfitting.
* F1-weighted is obtained by calculating F1 for each label, and finding their average weighted by support (the number of true instances for each label). This accounts for label imbalance. This may result in F1 score not falling between precision and recall.





**Inference**

* F1 weighted scores mean is 98.75%, which is very good. Boxplot of the F1 weighted score for all the five folds shows consistently good value , which is above 90%.

# SUPPORT VECTOR MACHINES (SVM)



## Introduction to SVM

* Support Vector Machine (SVM) is a supervised learning method that can be used for both classification and regression problems.
* The original support vector machines (SVMs) were invented by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963. They were designed to address an issue with logistic regression which places classification boundary arbitrarily close to a particular data point.
* SVM performs classification by finding the decision boundary or hyperplane that best divides a dataset into two classes. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall.
* **What is a Hyperplane?** Think it is line or a curve that linearly or non-linearly separates and classifies a set of data.
* **Support vectors** are the data points that lie closest to the decision boundary or hyperplane. In other words, the vectors that define the hyperplane are the support vectors. They are the data points most difficult to classify.



### How does SVM work?

* SVM works by dividing data into multiple classes using hyper-plane. In 2-dimensional space a hyper-plane is a line; 3-d space, hyper-plane is a plane and any space higher than 3D, it is called a hyper-plane.
* For creating a hyper-plane, SVM picks up two points (known as support vectors) that are closest to the hyper-plane and their distance from the hyper-plane must be identical. The hyper-plane with the greatest possible margin is chosen.
* What is a margin? The margin is a distance that separates all of the points in the data.
* In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using kernel tricks, implicitly mapping their inputs into high-dimensional feature spaces.

### Advantages and disadvantages of SVM

#### Advantages

* SVMs are very good when we have no idea on the data.
* SVM works well with even unstructured and semi structured data like text, Images and trees.
* The kernel trick is real strength of SVM. With an appropriate kernel function, we can solve any complex problem.
* Unlike in neural networks, SVM is not solved for local optima.
* It scales relatively well to high dimensional data.
* SVM models have generalization in practice, the risk of over-fitting is less in SVM.
* SVM is always compared with ANN. When compared to ANN models, SVMs give better results.

#### DISAdvantages

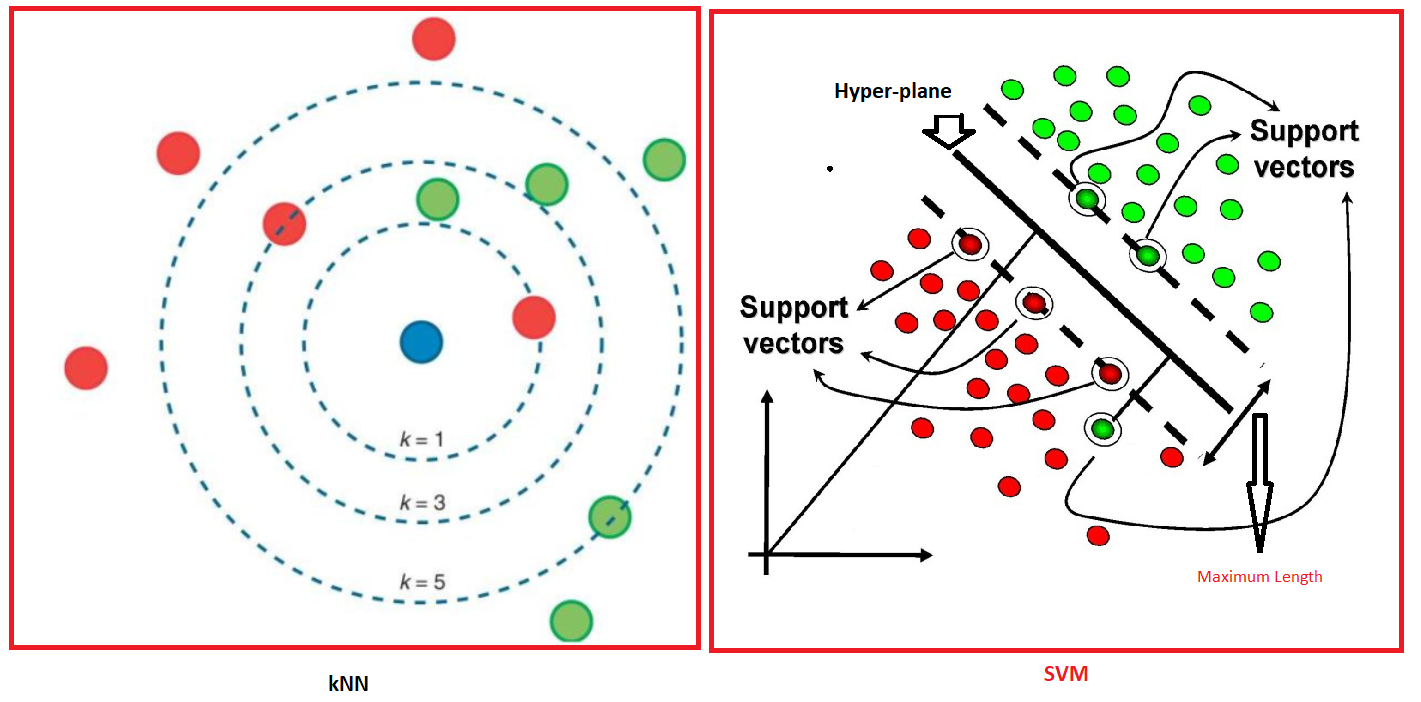
* Choosing a “good” kernel function is not easy.
* Long training time for large datasets.
* Difficult to understand and interpret the final model, variable weights and individual impact.
* Since the final model is not so easy to see, we can not do small calibrations to the model hence its tough to incorporate our business logic.
* The SVM hyper parameters are Cost -C and gamma. It is not that easy to fine-tune these hyper-parameters. It is hard to visualize their impact

### Applications of SVM

* Protein Structure Prediction
* Intrusion Detection
* Handwriting Recognition
* Detecting Steganography in digital images
* Breast Cancer Diagnosis
* Almost all the applications where ANN is used

### K Nearest Neighbors (KNN) Vs Support Vector Machines (SVM)

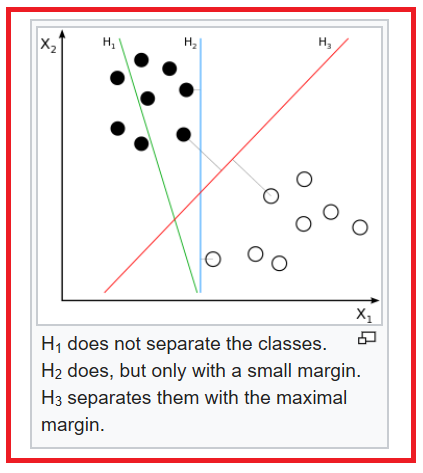
* kNN and SVM are the core algorithms used in Supervised machine learning especially in Optical Character Recognition (OCR).
* KNN is one of the simplest classification algorithms available for supervised learning and its idea is to search for closest match of the data in feature space. Thus, in kNN for a test data, we measure its distance to all the training samples and take the one with minimum distance. It is resource (time & memory) consuming to measure and store all the distances.
* KNN classifies data based on the distance metric whereas SVM need a proper phase of training.
* Generally, KNN is used as multi-class classifiers whereas standard SVM separate binary data belonging to either class. For a multiclass SVM, One-Vs-One, One-Vs-All approach is used.
* An SVM analyses data used for classification and regression analysis with the goal to find a line or hyper-plane that uniquely divides the data into two regions. For a test data, SVM finds a straight line or hyper-plane with largest minimum distance to the training samples. Using the training data, we find this decision boundary.
* Due to optimal nature of SVM, the data is optimally separated.
* Models trained by SVM can be used to predict for new data which are not labelled.
* For more details, refer to <https://www.ijraset.com/fileserve.php?FID=11852>



### SVM Classifiers

#### SVM Linear Classifier

* What is a linear classifier? In SVM, a data point is viewed as a p-dimensional vector (a list of p numbers), and we want to check if we can separate such points with (p - 1) dimensional hyper-plane. This is called a linear classifier. There may be many hyper-planes that classify the data. Best choice is the hyper-plane that represents the largest separation or margin, between the two classes. Such a hyper-plane is known as **maximum-margin hyper-plane** and the linear classifier it defines is known as a **maximum-margin classifier**, or equivalently, the perceptron of optimal stability.



Courtesy: https://en.wikipedia.org/wiki/File:Svm\_separating\_hyperplanes\_(SVG).svg

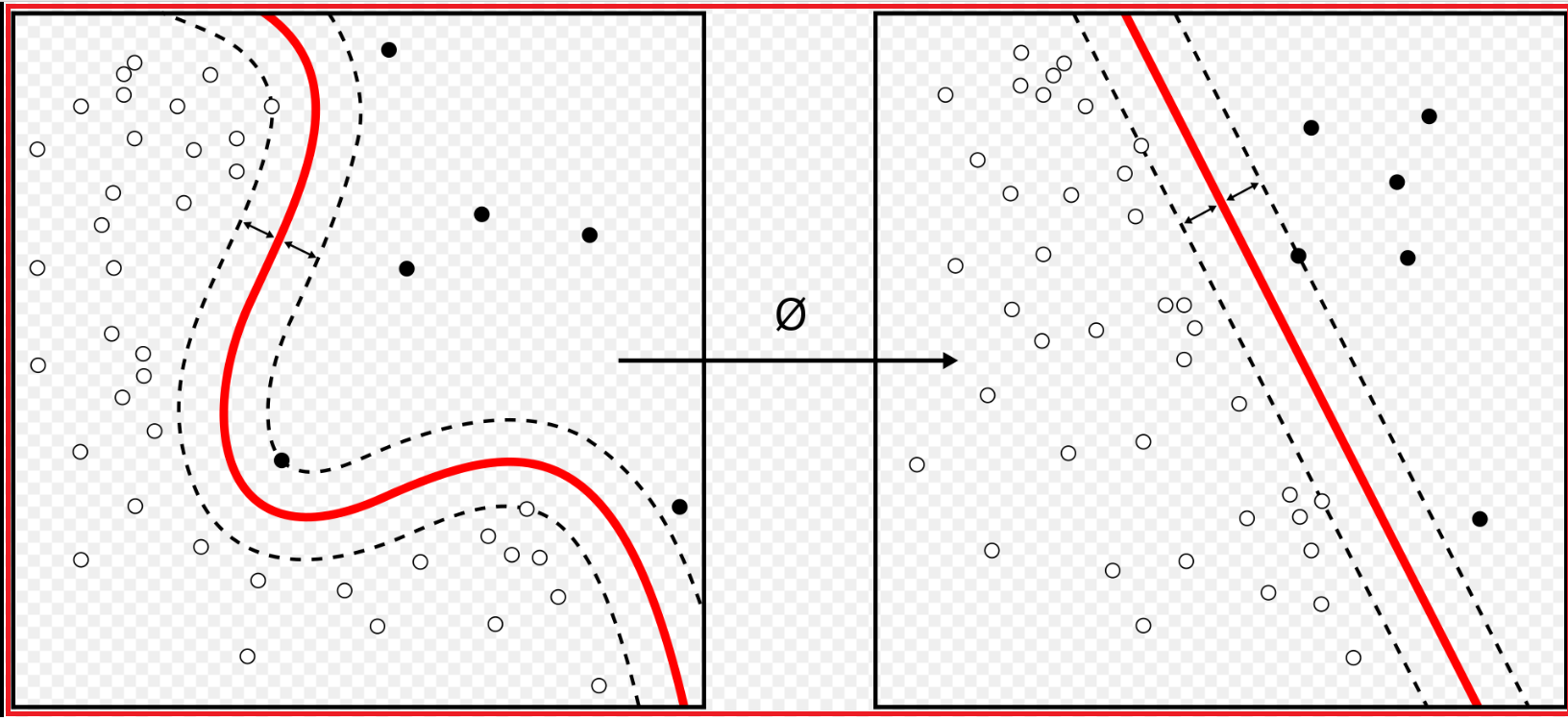
Refer: <https://dataaspirant.com/2017/01/13/support-vector-machine-algorithm/>

* Support vector machines in scikit-learn package provides various methods for supporting SVM linear classifiers:

1. SVC() - with parameter, kernel set to 'linear'
2. LinearSVC() without kernel parameter as this is assumed to be linear
3. NuSVC() (Nu Support vector classification) with parameter, kernel set to linear

#### SVM NON-Linear Classifier

* In 1992, Vapnik, Boser & Guyon suggested a way for building a non-linear classifier. They suggested using kernel trick in their research paper in 1995.
* In the real world, it may not be possible to separate data into different classes based on a straight linear hyper-plane.
* We need to create non-linear classifiers by applying the kernel trick to maximum-margin hyper-planes. In non-linear SVM classification, data points are plotted in a higher dimensional space.
* Kernel machines are used to compute non-linearly separable functions into a higher dimension linearly separable function.



* Courtesy: By Alisneaky, svg version by User:Zirguezi - Own work, CC BY-SA 4.0, <https://commons.wikimedia.org/w/index.php?curid=47868867>
* https://www.ijraset.com/fileserve.php?FID=3040

**What is a kernel trick?**

* Kernel trick is a method of using linear classifier to classify non-linear data points.
* Kernal is a function which performs the above task for us.

**Mercer's theorem**

* If a function K(x, y) satisfy all mercer's constraints, then there exists a function that maps x and y into higher dimension.

K(a, b) = Θ(x)T. Θ (y), where Θ () is a kernel.

* Using kernel trick, we can deal with any shape of data points.

**Popular kernel tricks**

The scikit-learn package has a method SVC() with the argument, kernel which specifies any of the following type to be used in the algorithm:

1. **Linear**

Linear kernel equation is given below:

K(x, x') = xTx' (A simple dot product)

* Linear kernel is good when we have a lot of features and faster than others. It is well suited for linear models such as text categorization.

1. **poly for polynomial**

K(x, x') = (ϒ. xTx' + r)d, ϒ > 0

Here d and ϒ are kernel parameters

* Polynomial kernel represents the similarity of vectors (training samples) in a feature space over polynomials of the original variables allowing learning of non-linear models.
* Refer <https://en.wikipedia.org/wiki/Polynomial_kernel>.

1. **rbf (default)**

* Radial Basis Function kernel (RBF kernel) is a popular kernel function used in various kernelized learning algorithms such as SVM classification.
* RBF is a general-purpose kernel and used when there is no prior knowledge about the data.
* The RBF kernel on two samples x and x', represented as feature vectors in some input space is defined as

K(x, x') = exp( - ϒ ||x - x' ||2) This is a Gaussian Radial Basis function and

where ϒ= 1 / (2σ2) and ϒ > 0

||x - x' ||2 is the squared Euclidean distance between two data points x and x'

and ϒ is a parameter of the RBF kernel and can be thought of as the spread of the kernel and hence the decision region.

* When the ϒ is low, the curve of the decision boundary is very low and thus the decision region is very broad.
* When the ϒ is high, the curve of the decision boundary is high and it creates several decision boundaries around the data points.
* Since the value of the RBF kernel decreases with distance and ranges between 0 and 1, this can be easily interpreted as a similarity measure.
* Refer: <https://en.wikipedia.org/wiki/Radial_basis_function_kernel>

1. **Sigmoid**

* Sigmoid kernel is used for neural networks.
* Sigmoid kernel equation is given below:

K(x, x') = tanh(ϒ xTx' + c) for c < 0

where ϒ and c are kernel parameters

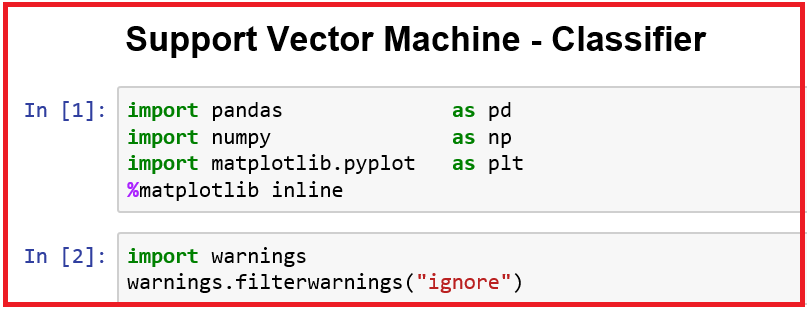
* Refer
* https://medium.com/@ankitnitjsr13/math-behind-svm-kernel-trick-5a82aa04ab04
* https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html
* https://brilliant.org/wiki/support-vector-machines/
* <https://en.wikipedia.org/wiki/Support-vector_machine>
* <https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72>
* <https://www.saedsayad.com/support_vector_machine.htm>
* <https://deepai.org/machine-learning-glossary-and-terms/support-vector-machine>
* https://techwithtim.net/tutorials/machine-learning-python/svm-2/

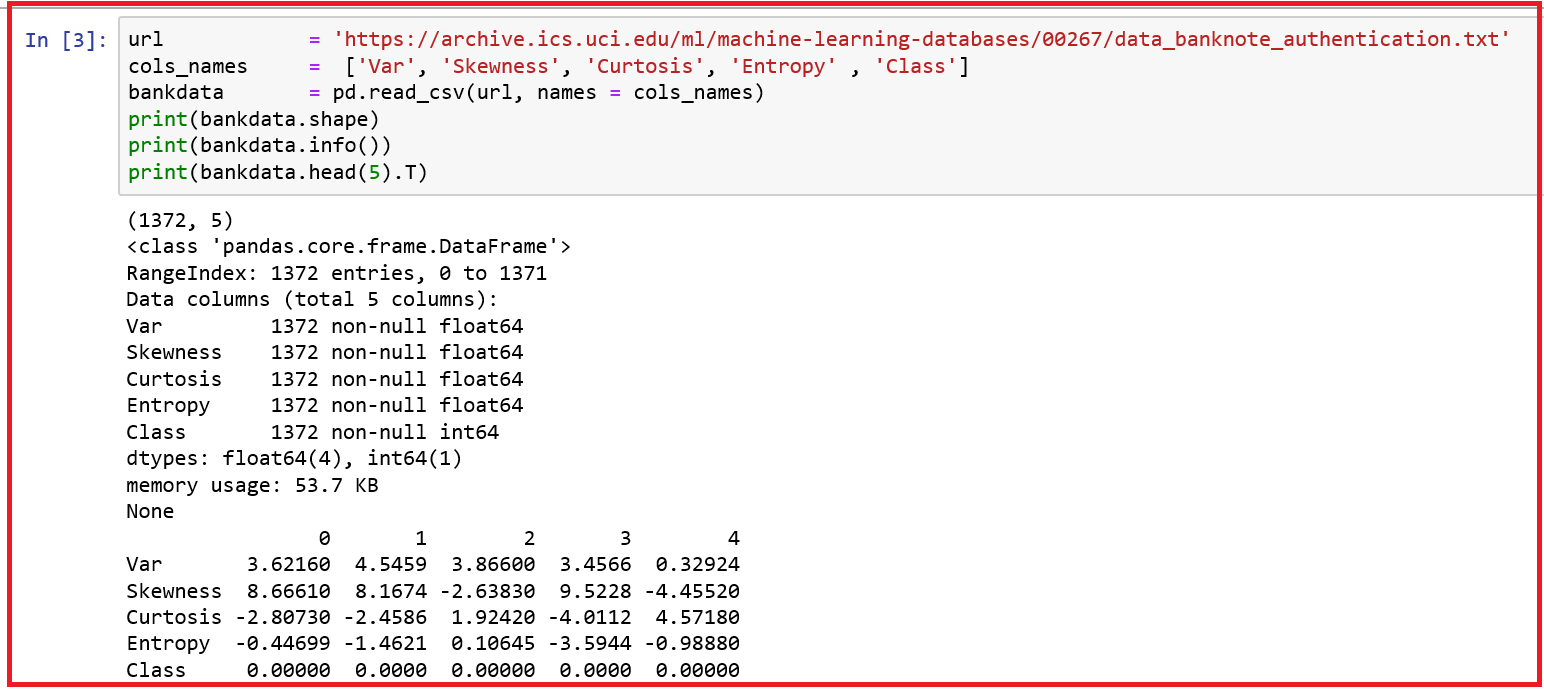
#### SUPPORT VECTOR CLASSIFICATION EXAMPLE

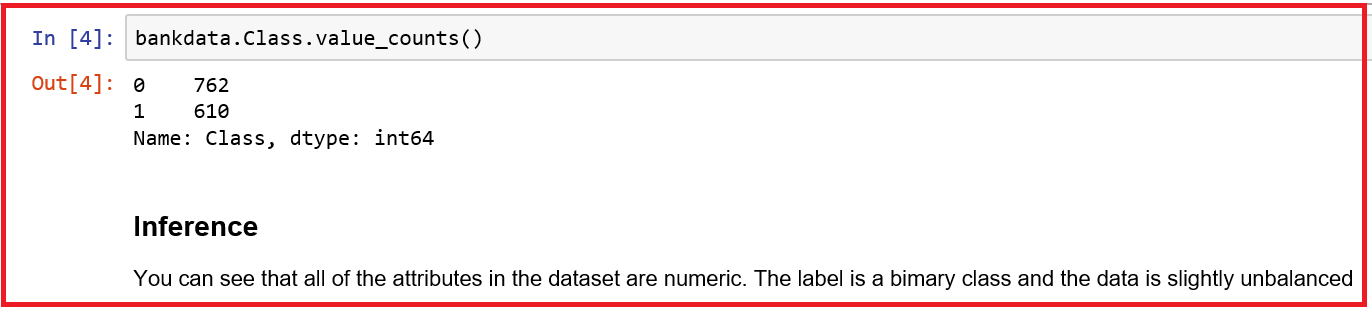
**Data**

* Data were extracted from images that were taken from genuine and forged banknote-like specimens. For digitization, an industrial camera usually used for print inspection was used. The final images have 400x 400 pixels. Due to the object lens and distance to the investigated object gray-scale pictures with a resolution of about 660 dpi were gained. Wavelet Transform tool were used to extract features from images.
* Source: Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [[http://archive.ics.uci.edu/ml]](http://archive.ics.uci.edu/ml%5D). Irvine, CA: University of California, School of Information and Computer Science.
* We shall use SVM classification technique for classify the images.

**Load required packages:**



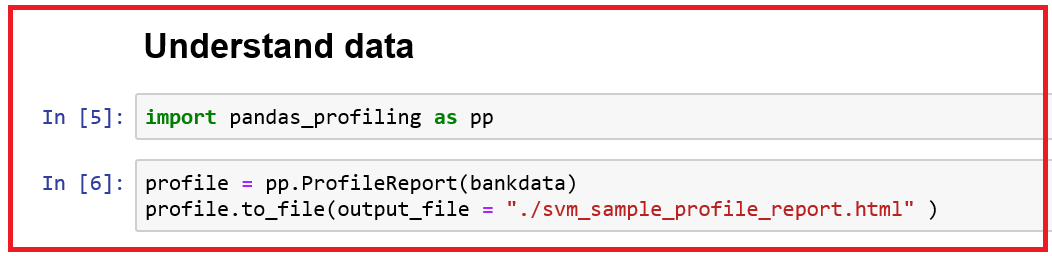


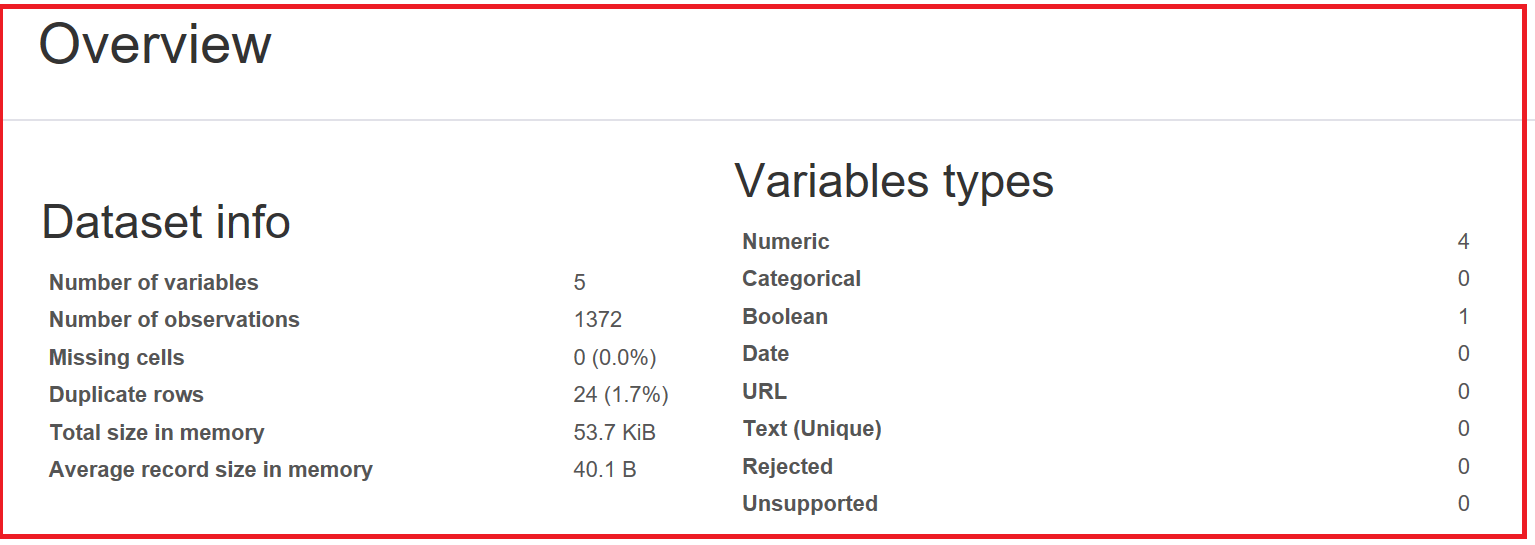


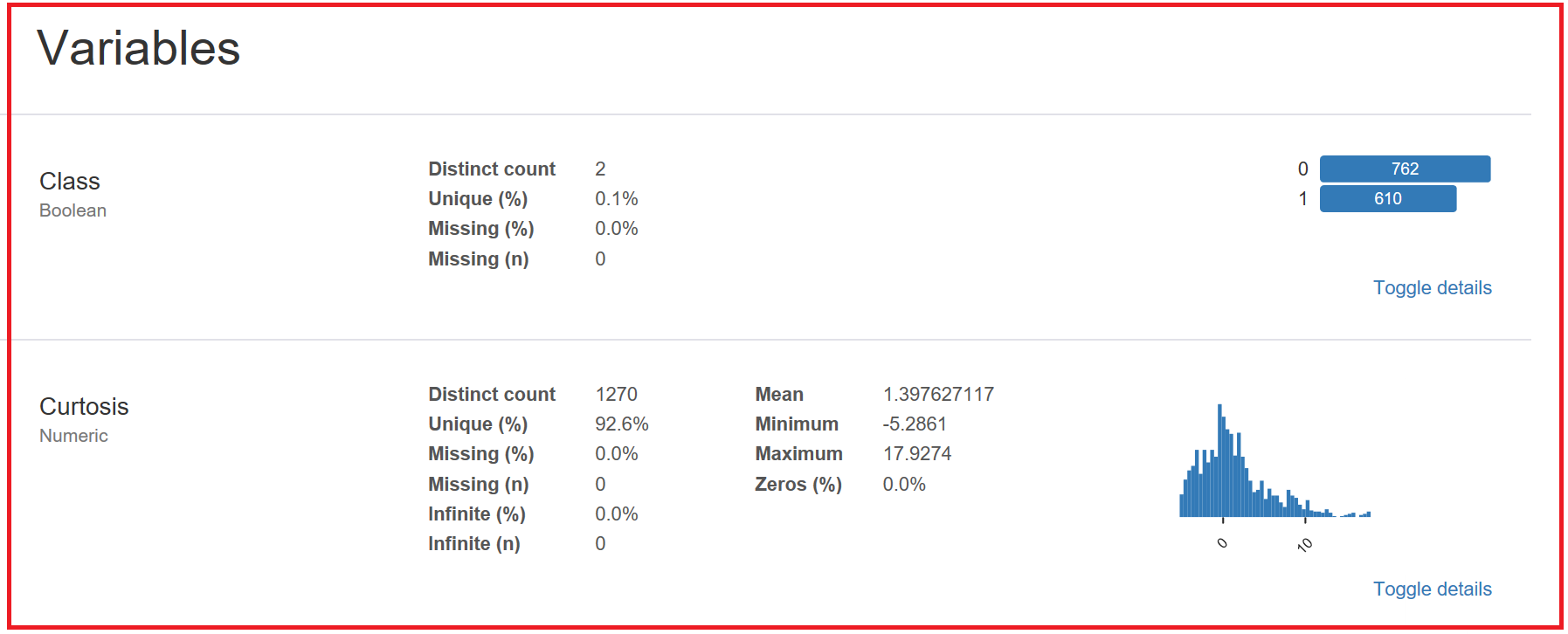
**Quick EDA using pandas-profiling**

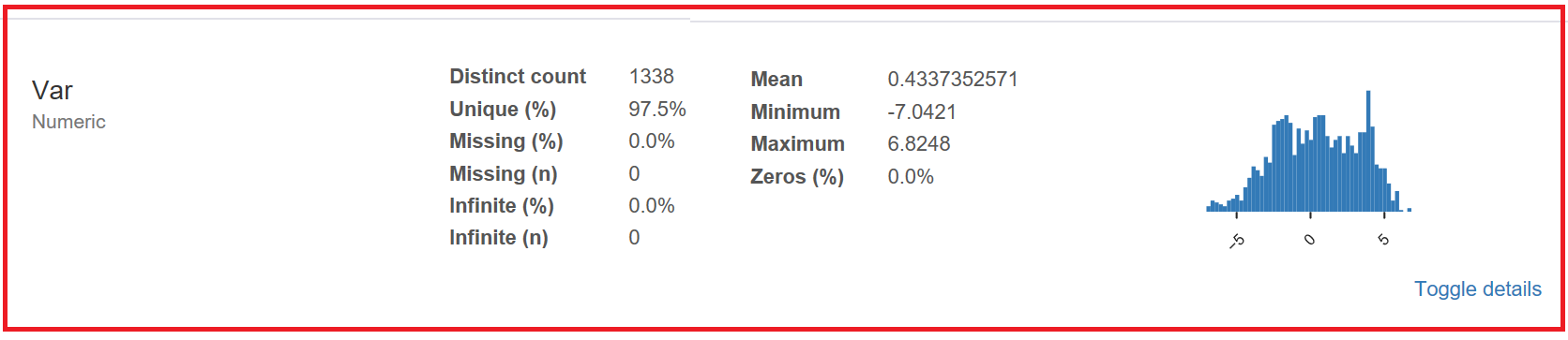
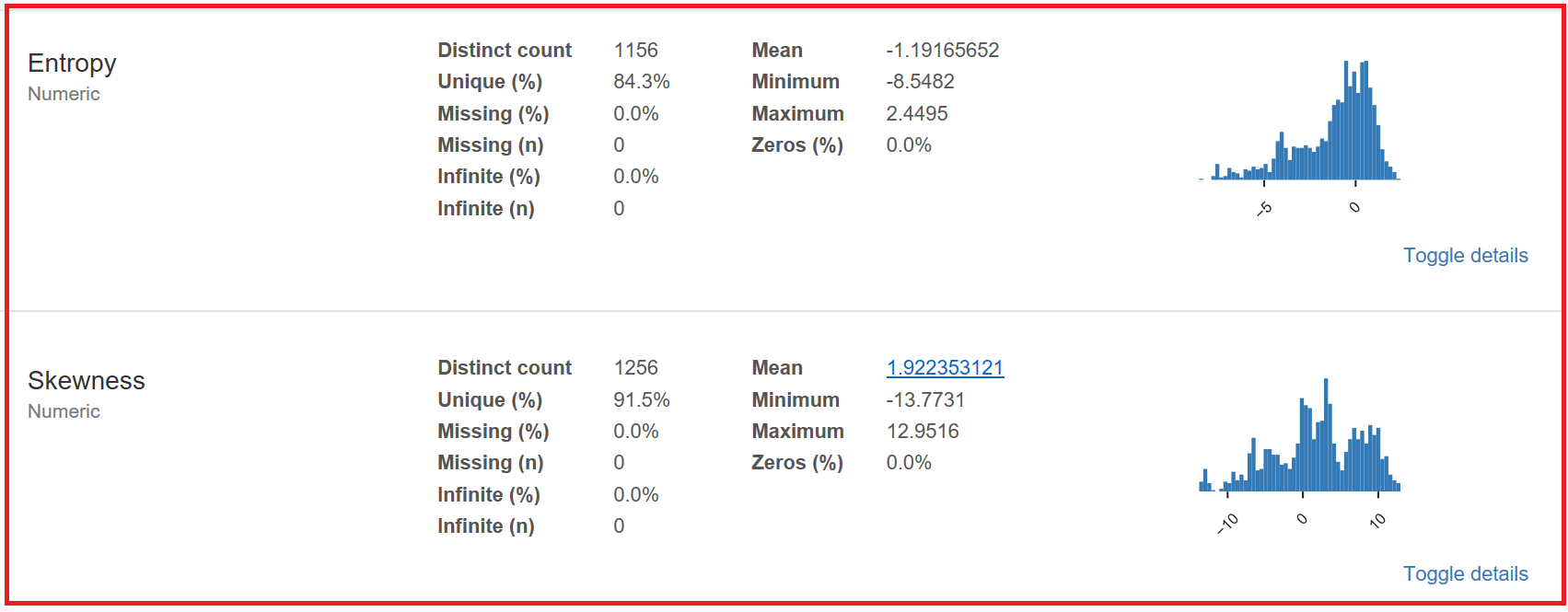
* The pandas-profiling Python package is a great tool to create HTML profiling reports. For a given dataset, it computes the following statistics:

1. Essentials: type, unique values, missing values.
2. Quantile statistics like minimum value, Q1, median, Q3, maximum, range, interquartile range.
3. Descriptive statistics like mean, mode, standard deviation, sum, median absolute deviation, coefficient of variation, kurtosis, skewness.
4. Most frequent values.
5. Histogram.
6. Correlations highlighting of highly correlated variables, Spearman and Pearson matrixes.

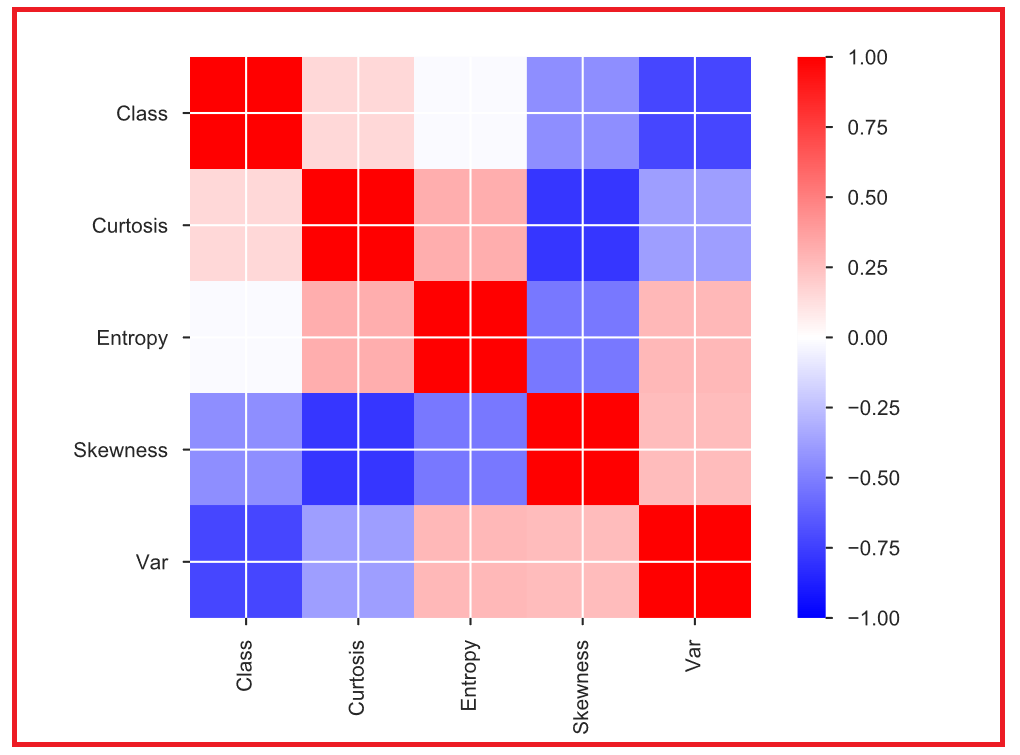


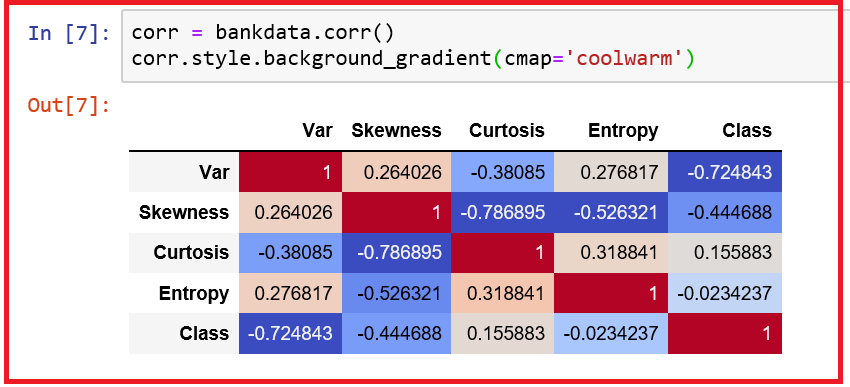






**Correlations**

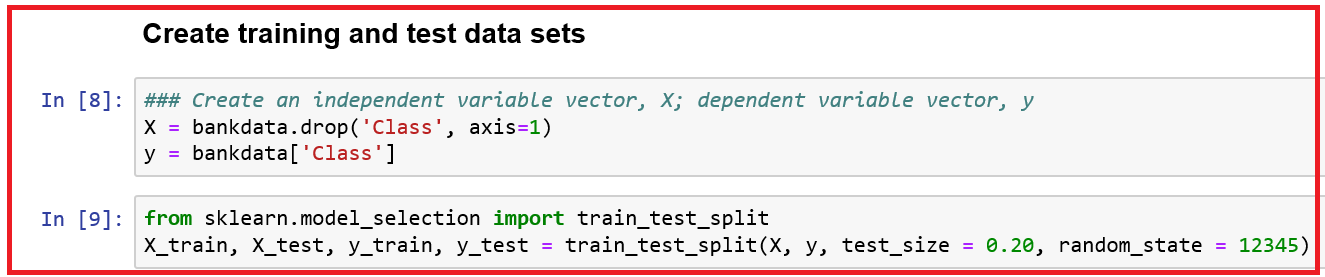




* *From the above plot of correlation coefficients, we observe that correlation coefficient is high between skewness and Curtosis (-0.79), Skewness and Entrophy (-0.53) among the independent variables.*

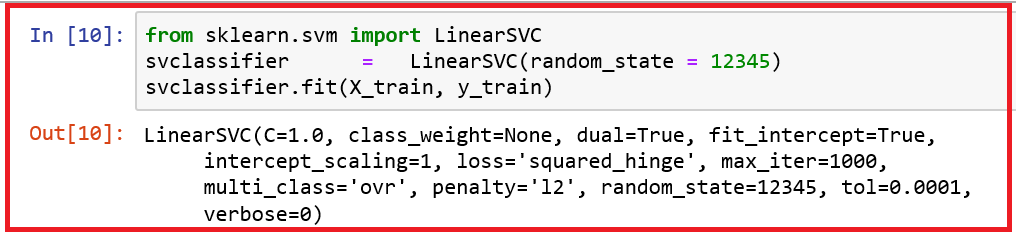
|  |  |
| --- | --- |
|  |  |
|  |  |

**Create two data sets - X for independent variables and y for dependent variable**



**Train the algorithm**

* *Scikit-learn contains svm package, which contains built-in classes for different SVM algorithms. We are interested in performing a classification task.*
* *We shall use the support vector classifier class, LinearSVC, a simple SVM that classifies linearly separable data.*
* *The fit method of SVC class is called to train the algorithm on the training data, which is passed to the parameter of the fit method.*



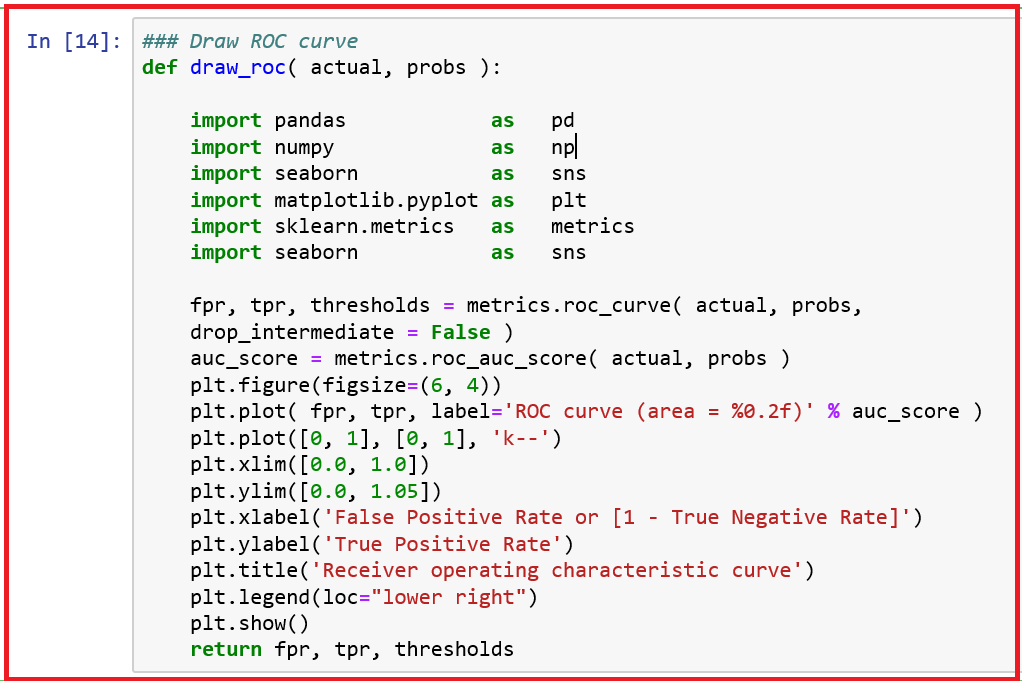
**Predictions**

* *Use the predict method of the LinearSVC class.*

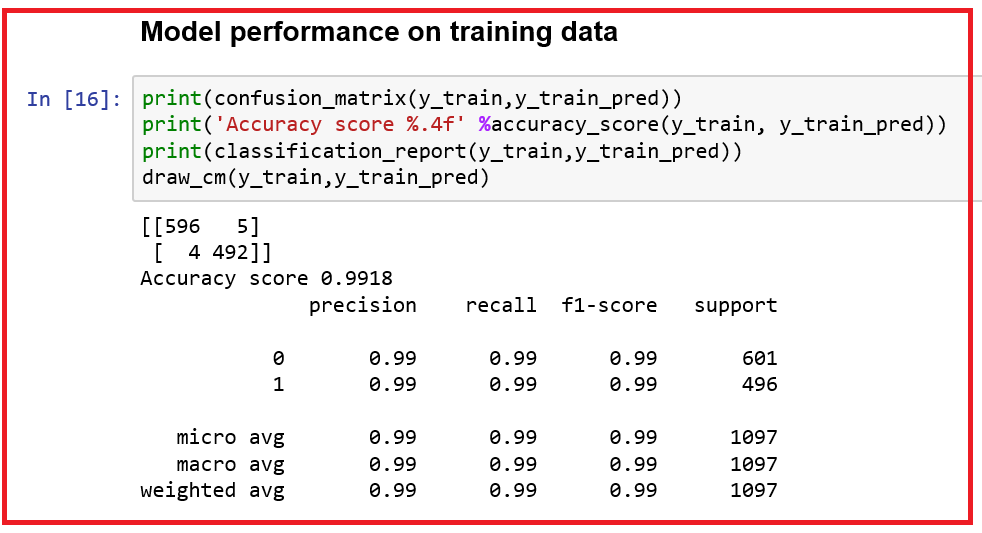
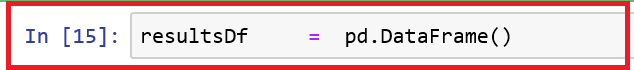


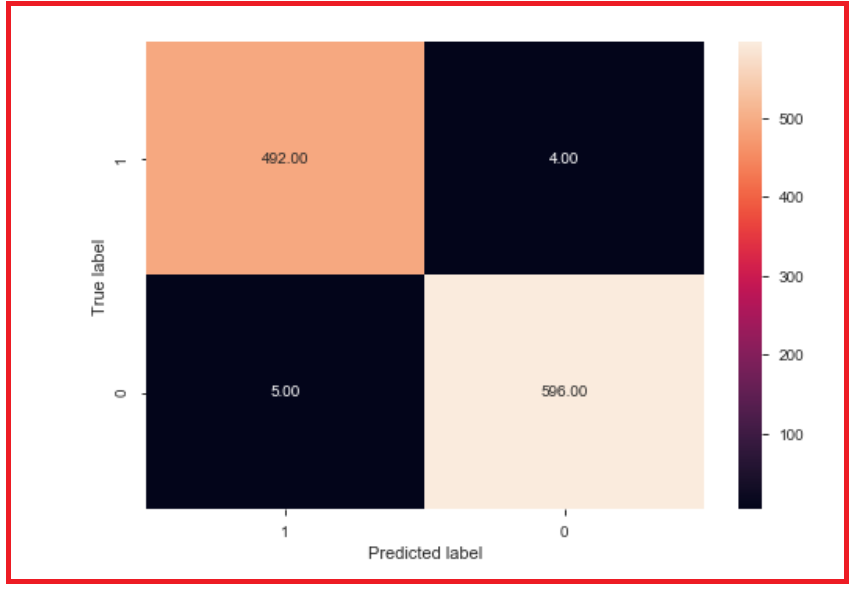
**Evaluate Model performance**

* *We shall use both classification\_report and confusion\_matrix methods available in the Scikit-learn metrics library to get various accuracy measures such as confusion matrix, precision, recall and F1 score.*
* *We shall import necessary modules and also define useful functions.*

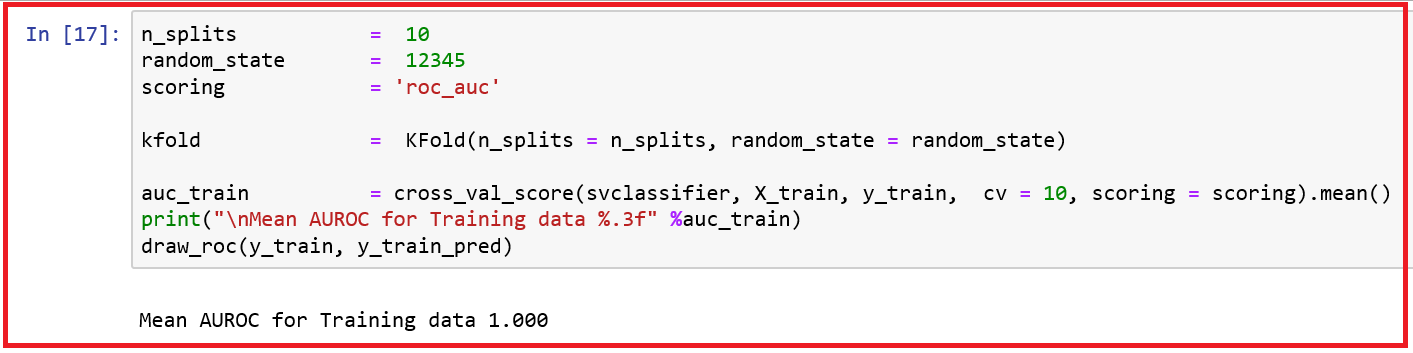


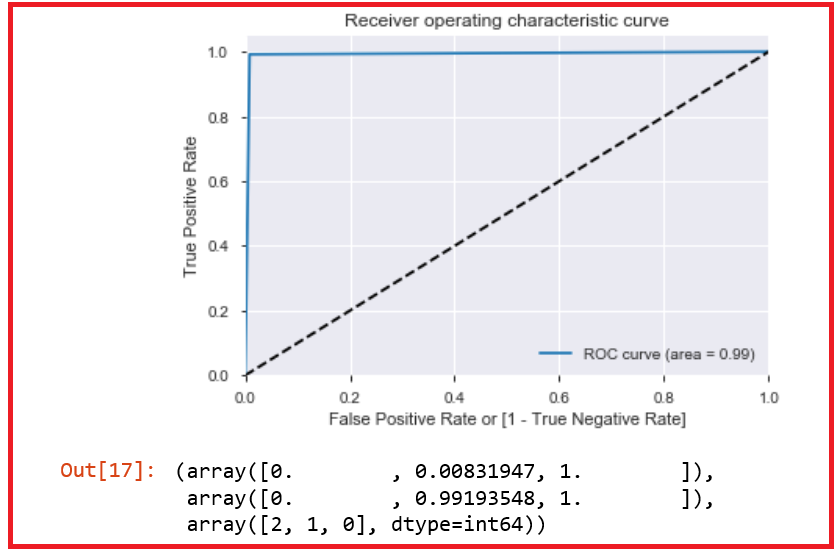
* *We shall create an empty data frame to hold the evaluation metrics.*



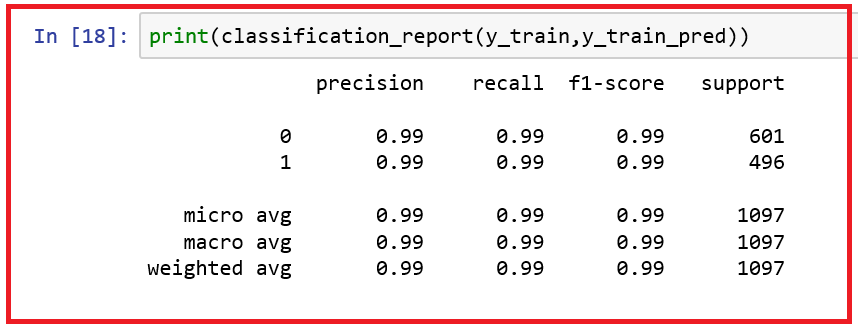


* *There are nine misclassifications in the case of SVM Linear algorithm. Accuracy ratio for the training data set is 0.9918.*
* *Compute mean AUROC after creating ten-fold cross validation.*

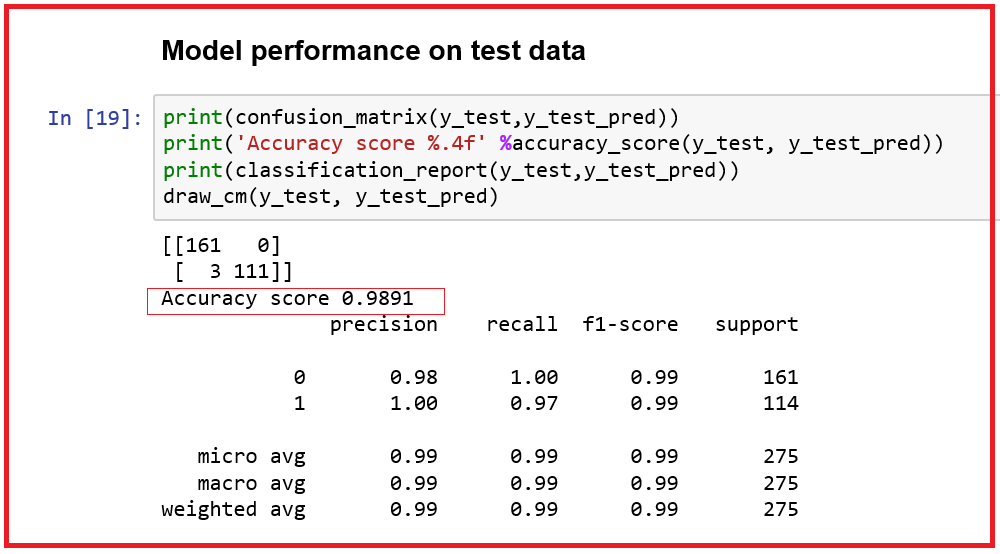
**

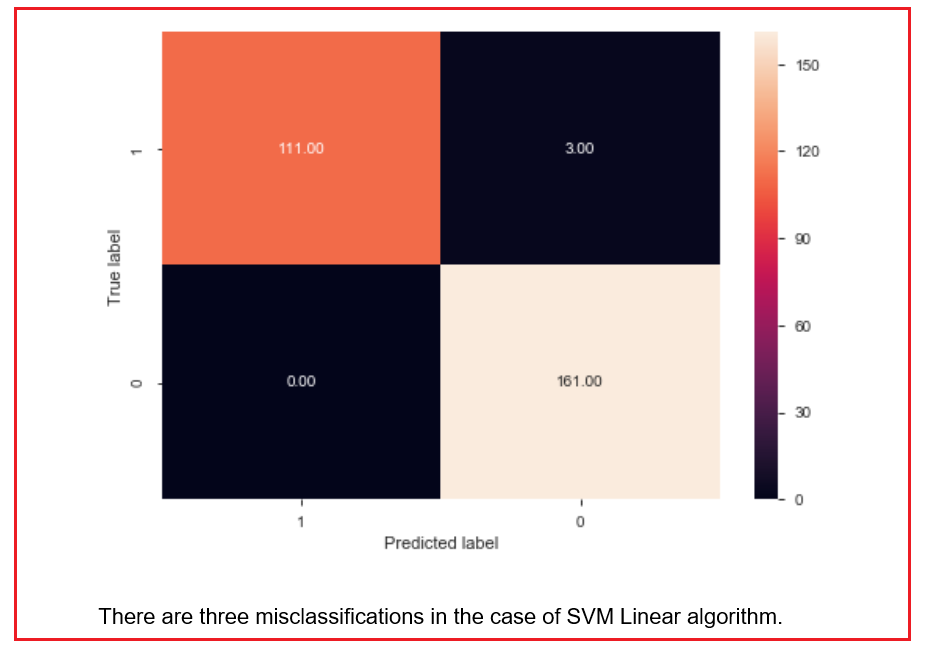
**

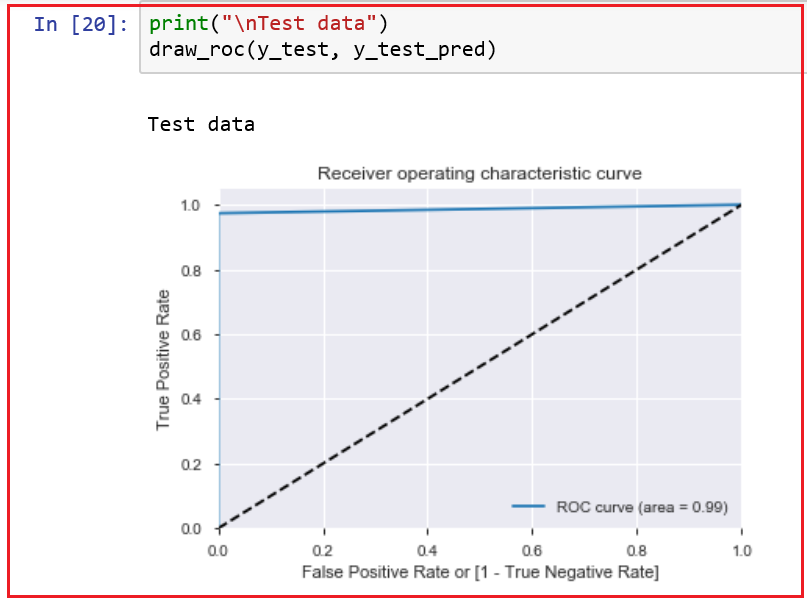
* *We observe that the computed mean AUROC is 1 (too good to be true). We will go ahead with computing other measures.*

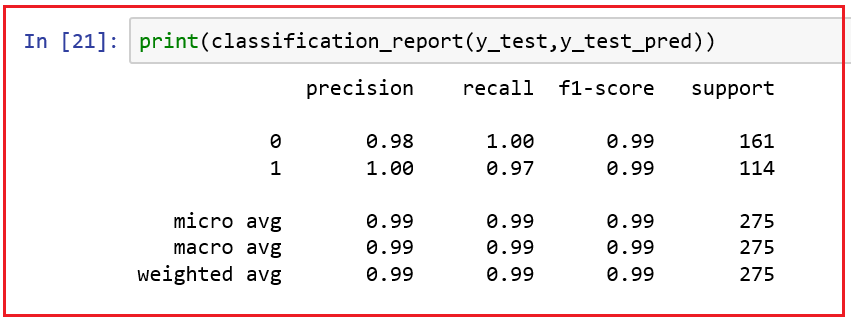
**

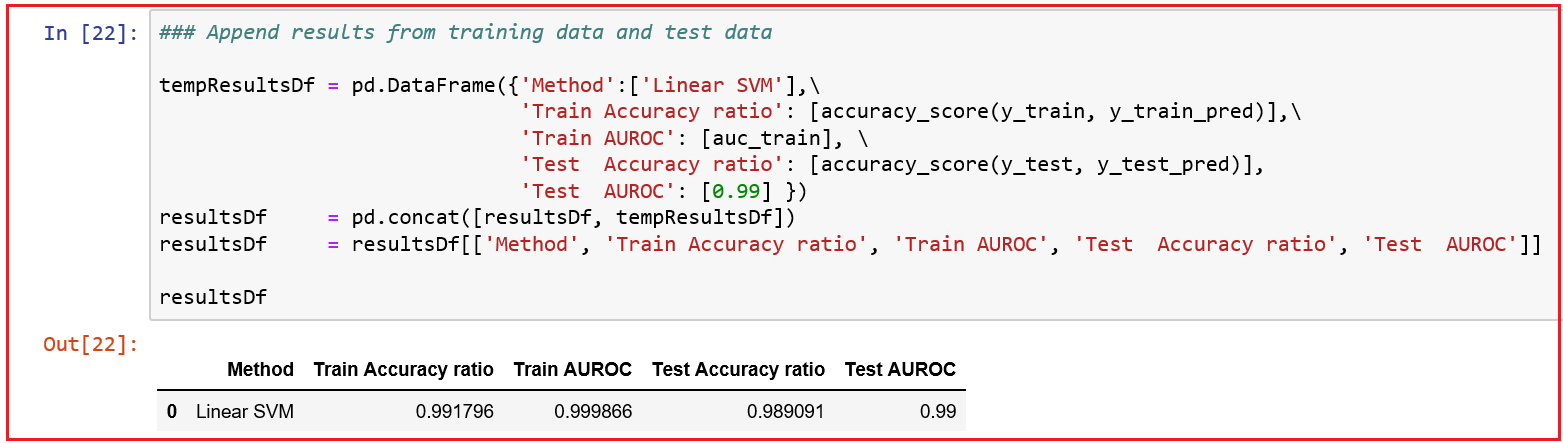
* *We observe that the Precision, recall and F1 scores for both ‘0’ and ‘1’ classes are 0.99.*
* *Let us compute the above measures for the test data and check how the model performs.*







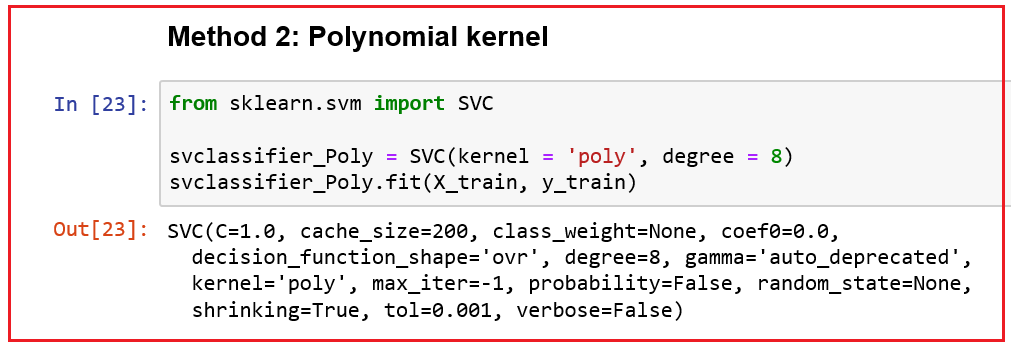
**

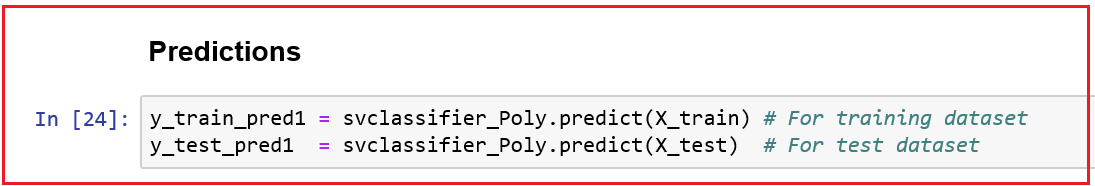
**

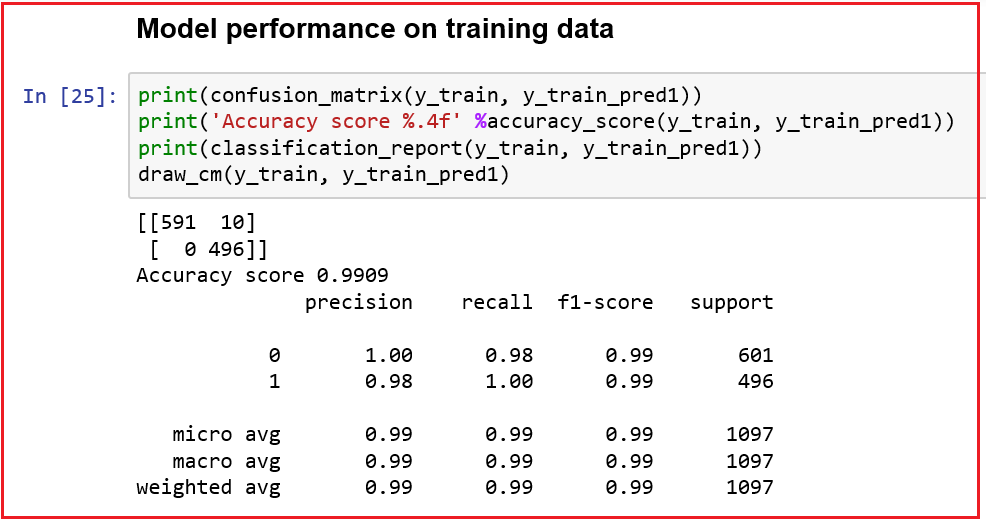
* *We observe that there is no performance degradation between training and test data and there is no overfit of the model.*

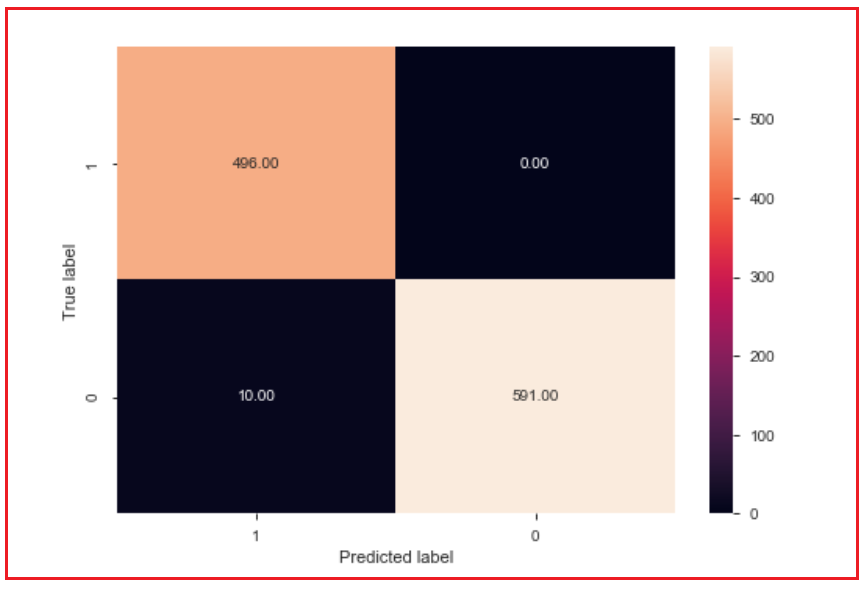
***Try Kernel tricks***

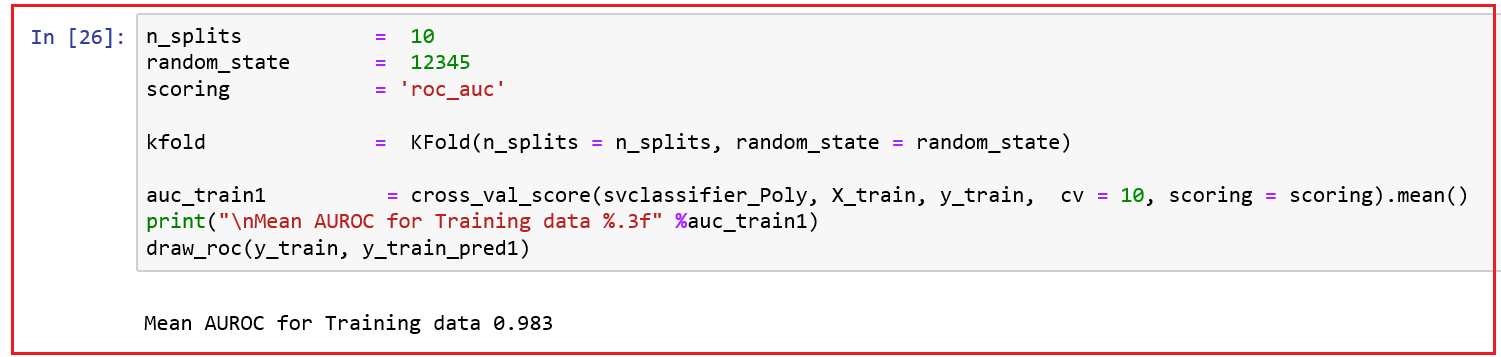
* *First try polynomial kernel trick with 8 degrees.*

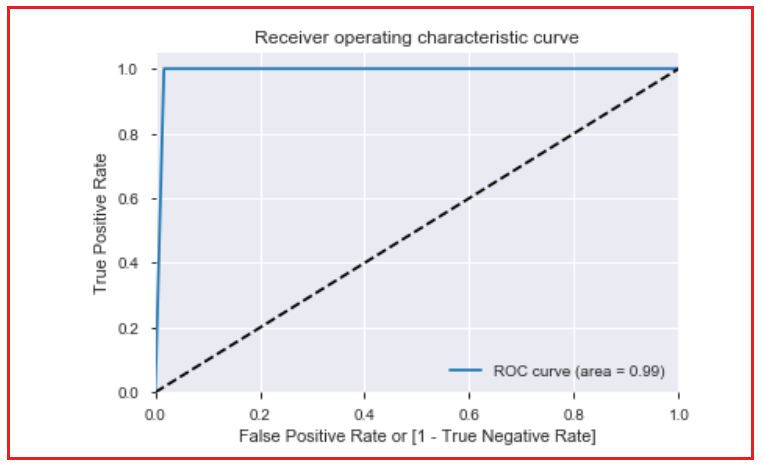
**

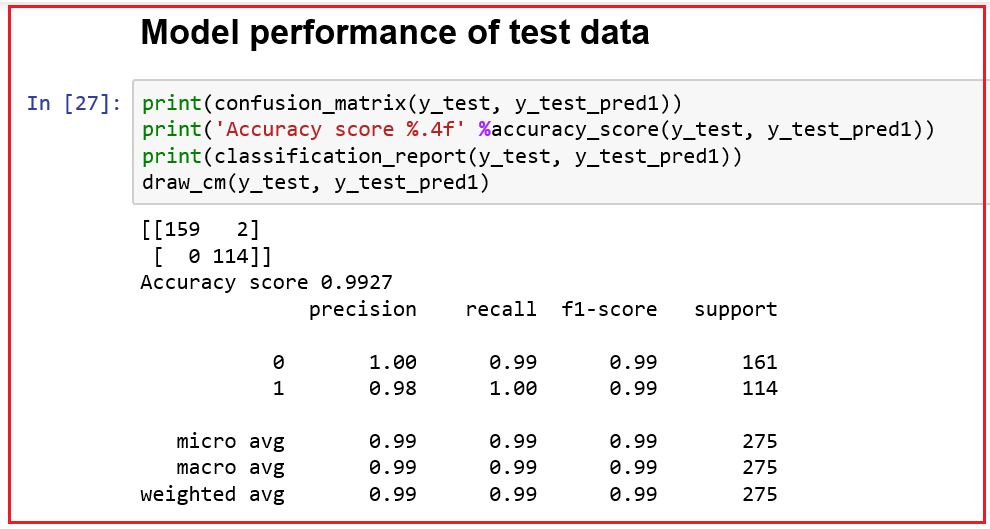
**

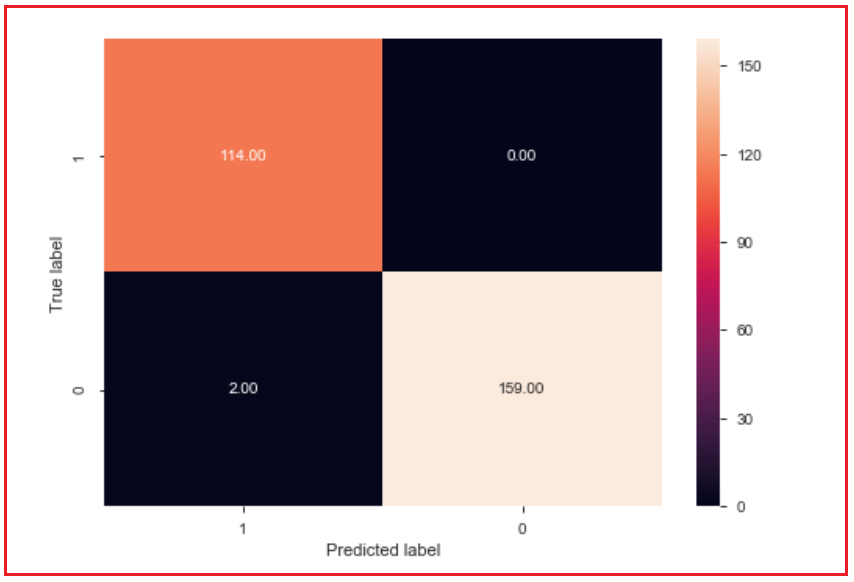
**

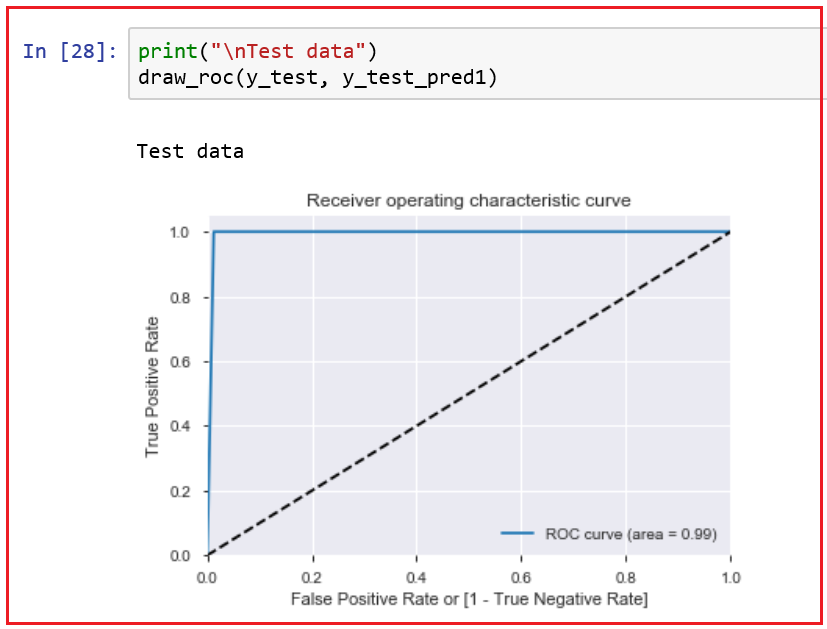
**

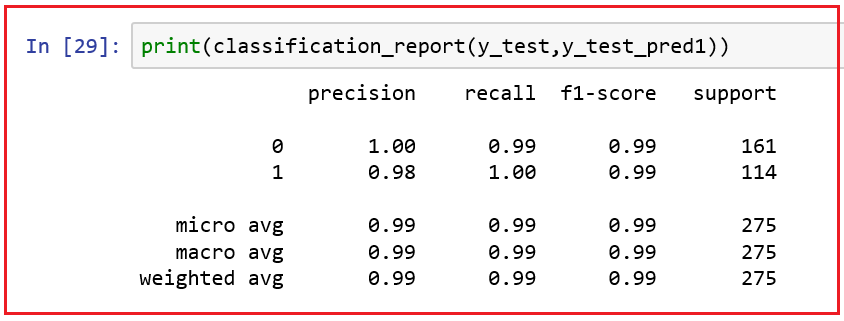
**

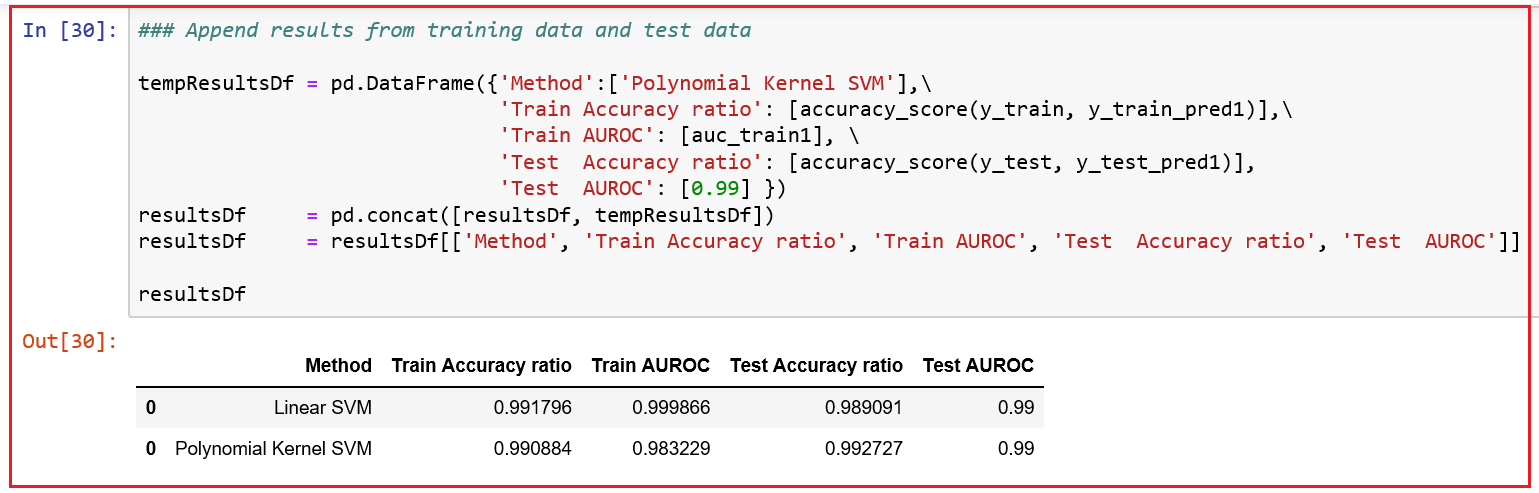
**

**

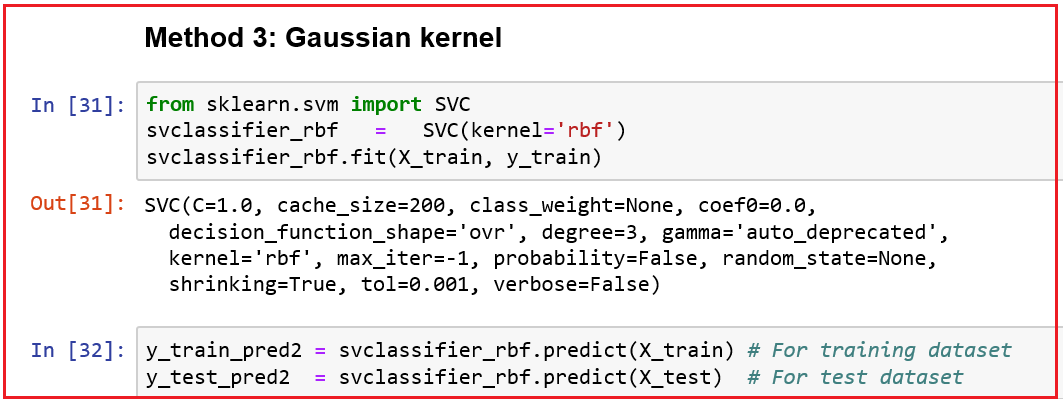
**

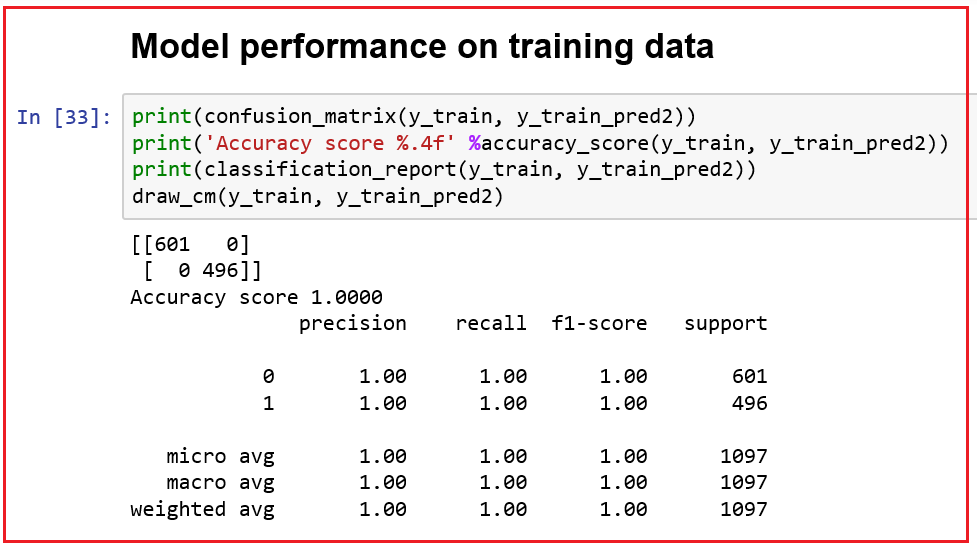
**

**

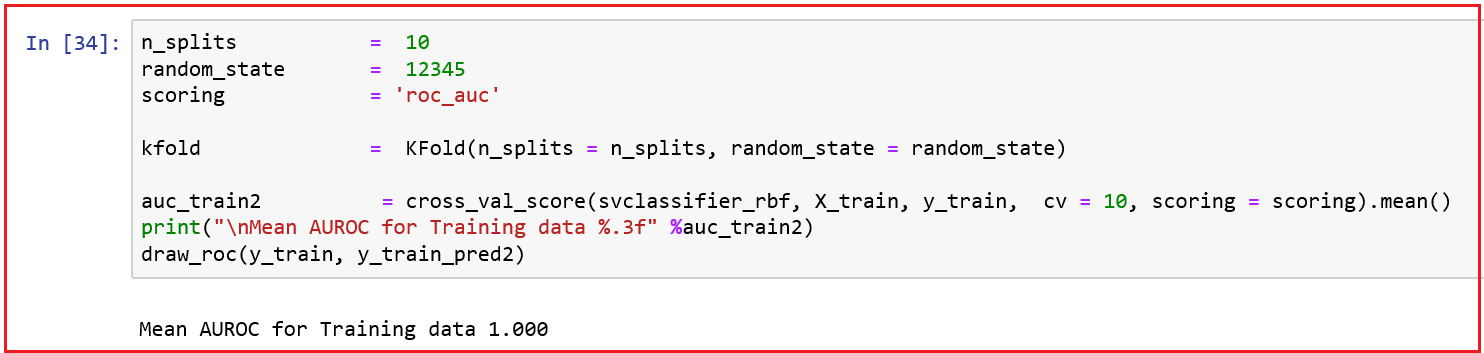
**

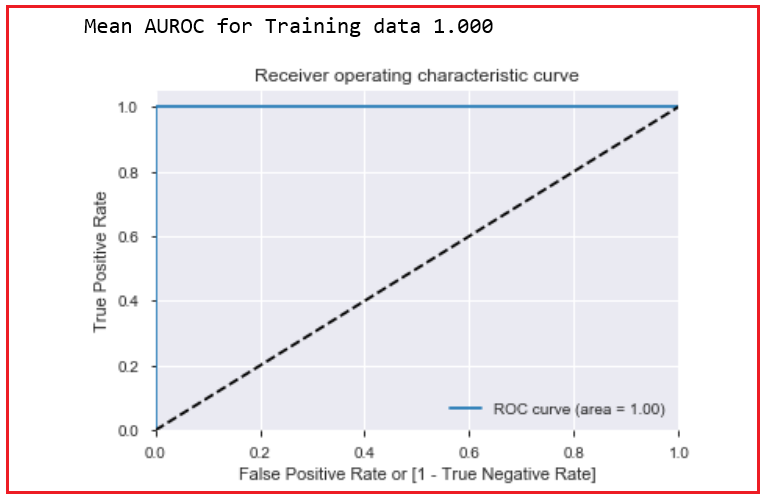
* *We observe that the Polynomial kernel trick has performed better than Linear SVM. Number of mis-classifications is reduced from 3 to 2 for test data in case of Polynomial SVM when compared to Linear SVM.*
* *Let us try another kernel trick, ‘rbf’*

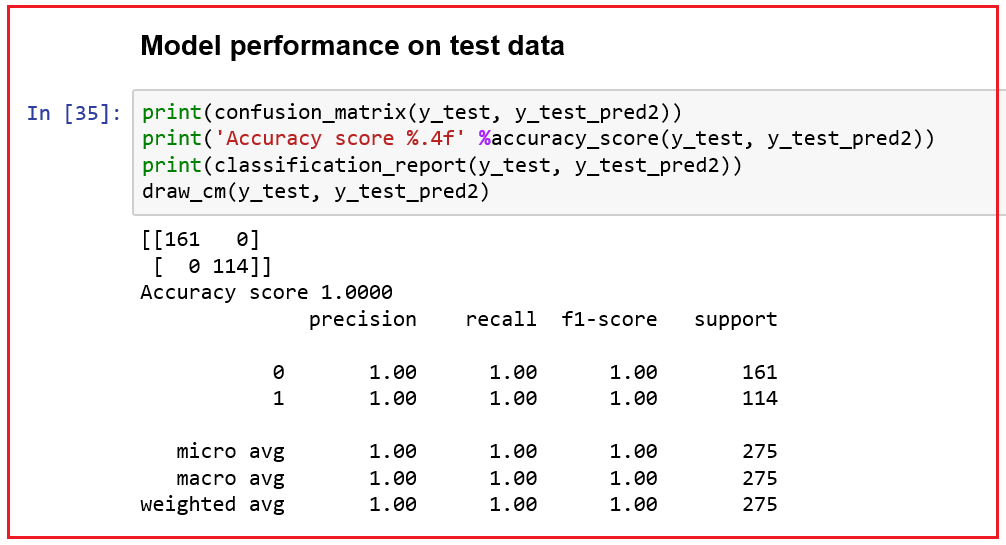
**

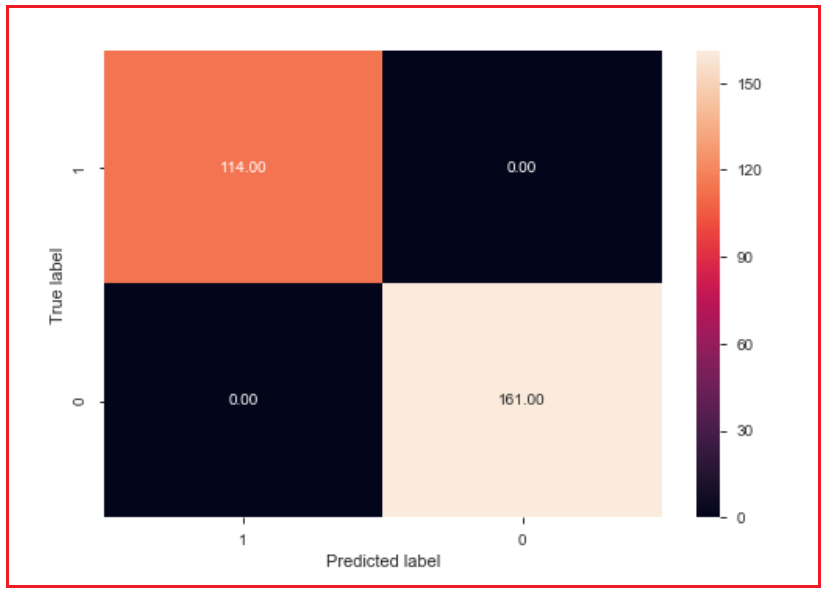
**

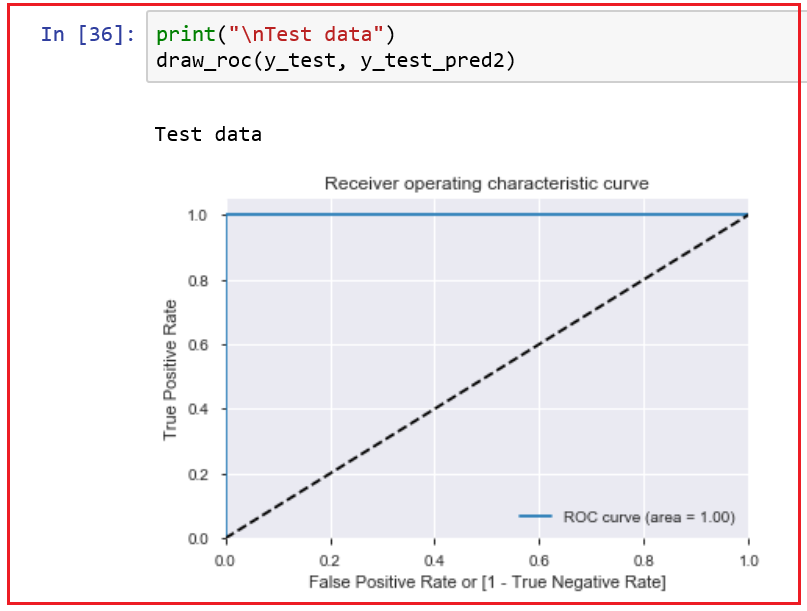
**

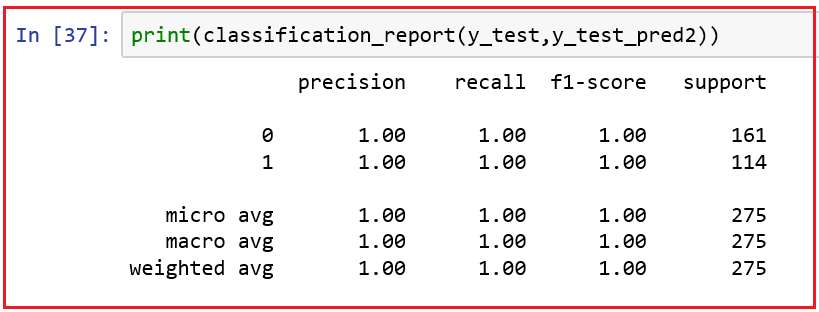
**

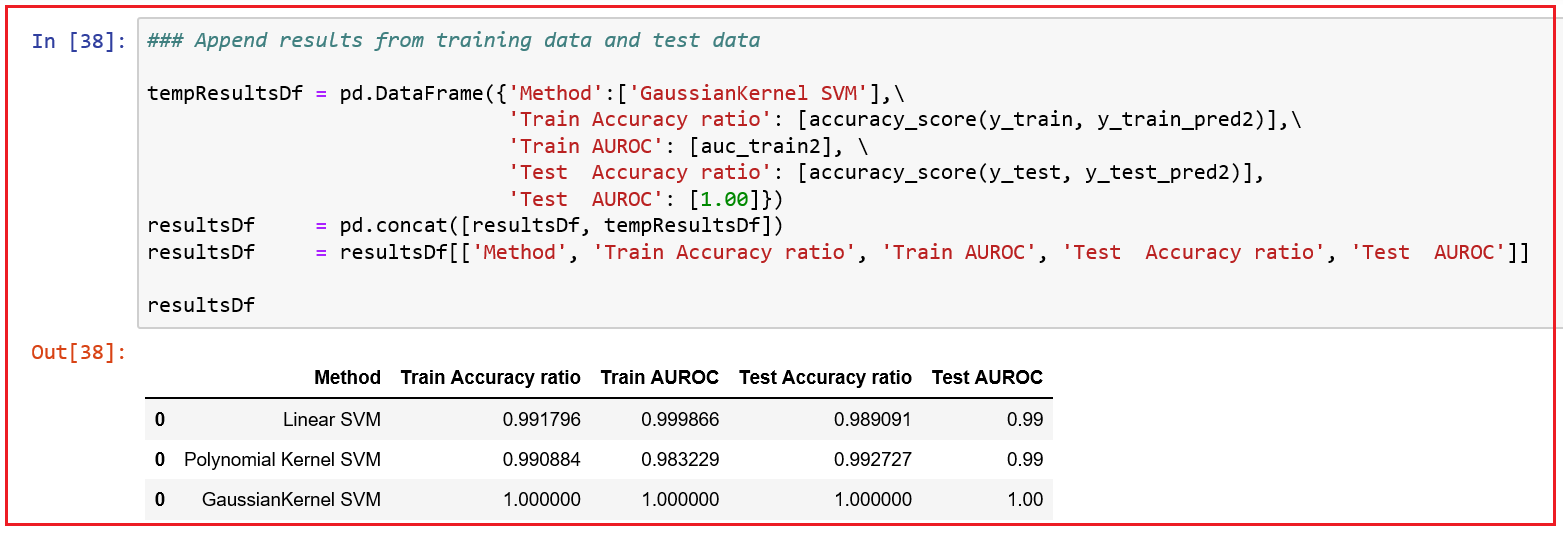




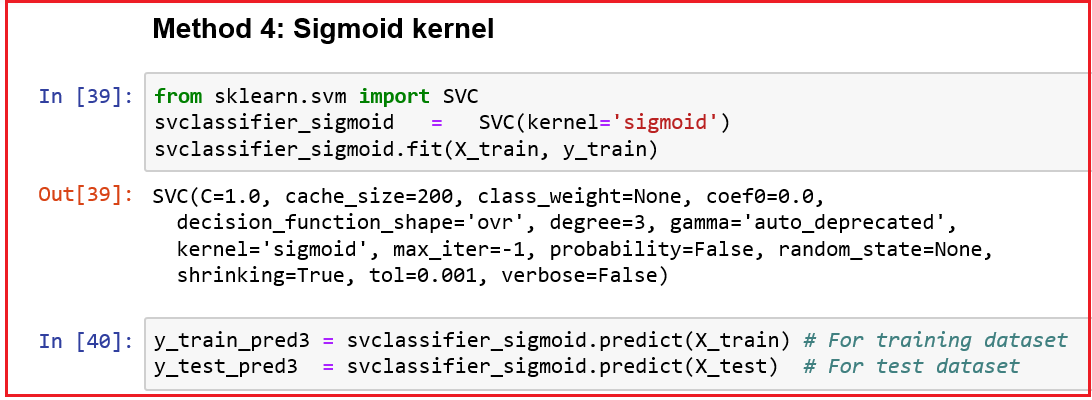


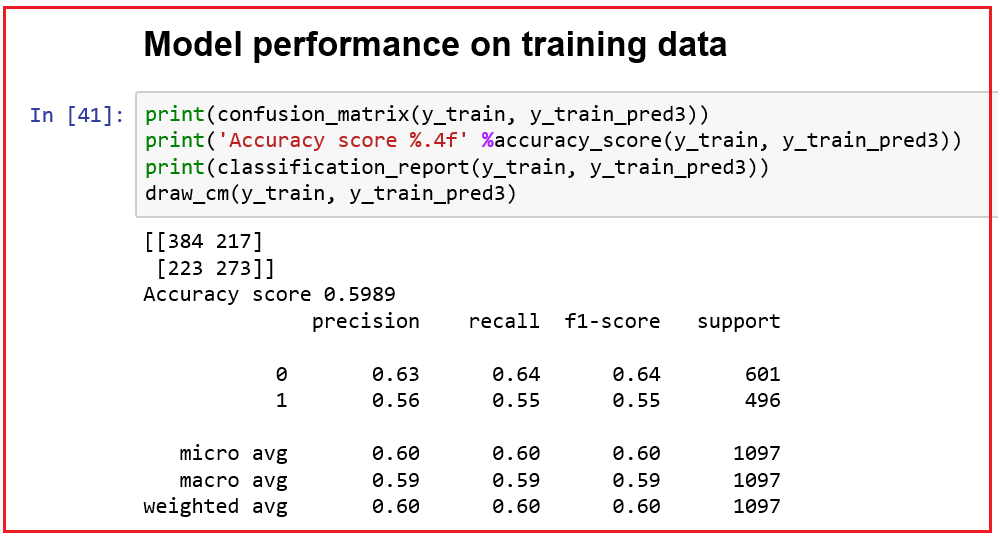


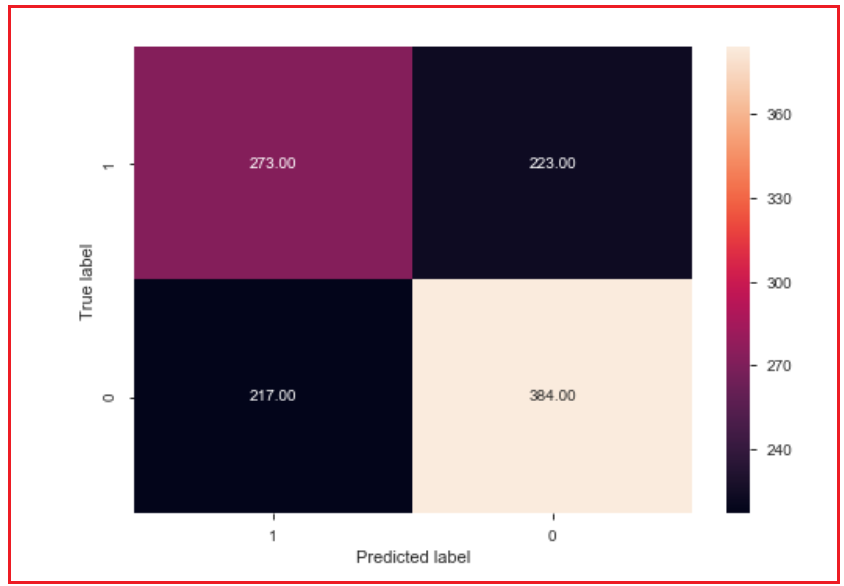


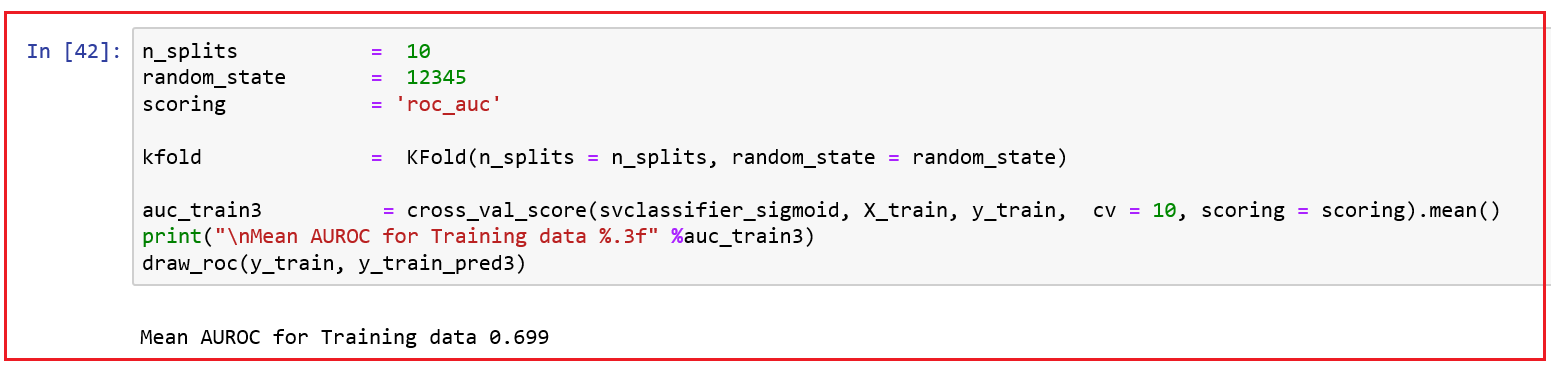


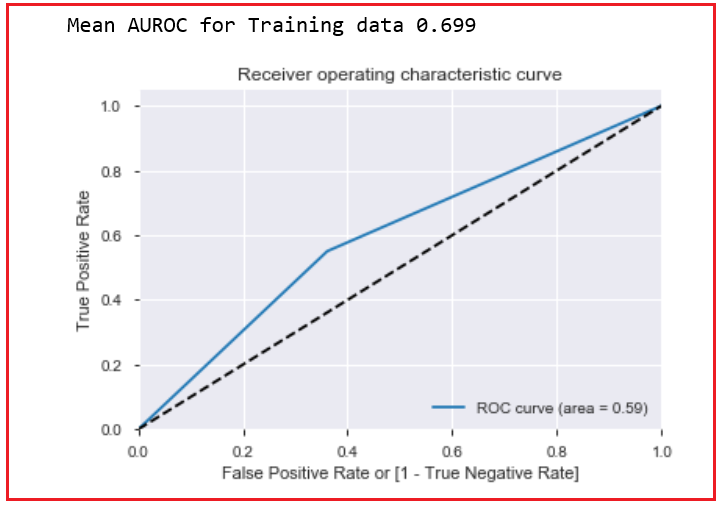
* *We observe that the Gaussian RBF kernel trick has performed better than both Polynomial SVM and Linear SVM. Number of mis-classifications is reduced from 2 to 0 for test data when compared to Polynomial.*
* *Let us try another kernel trick, ‘Sigmoid’*

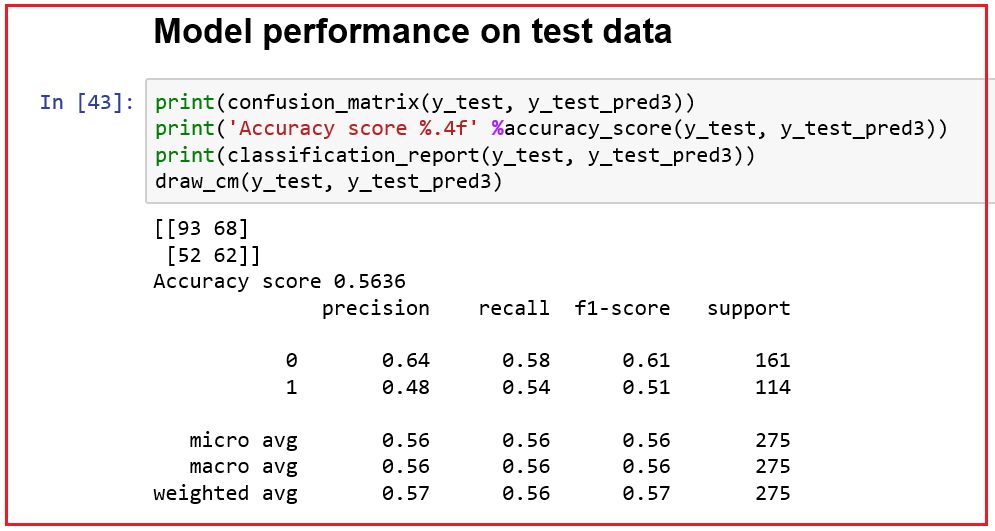
**

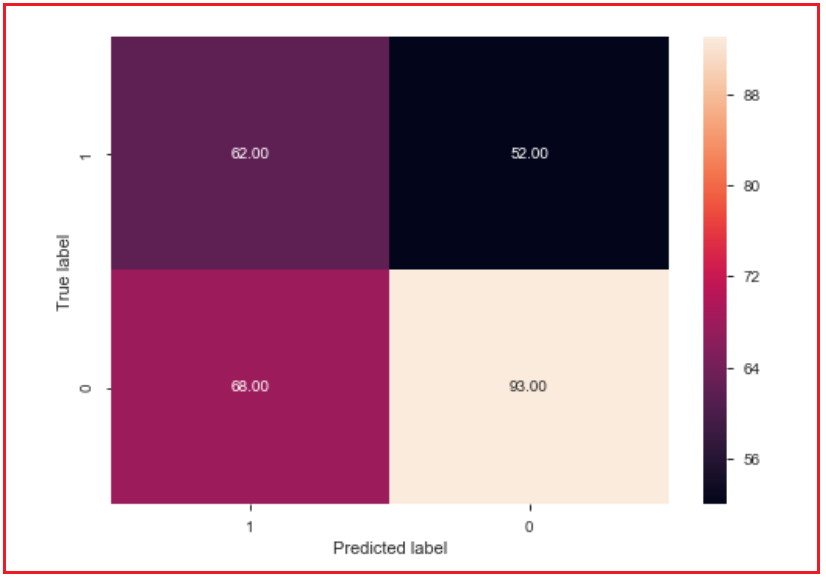
**

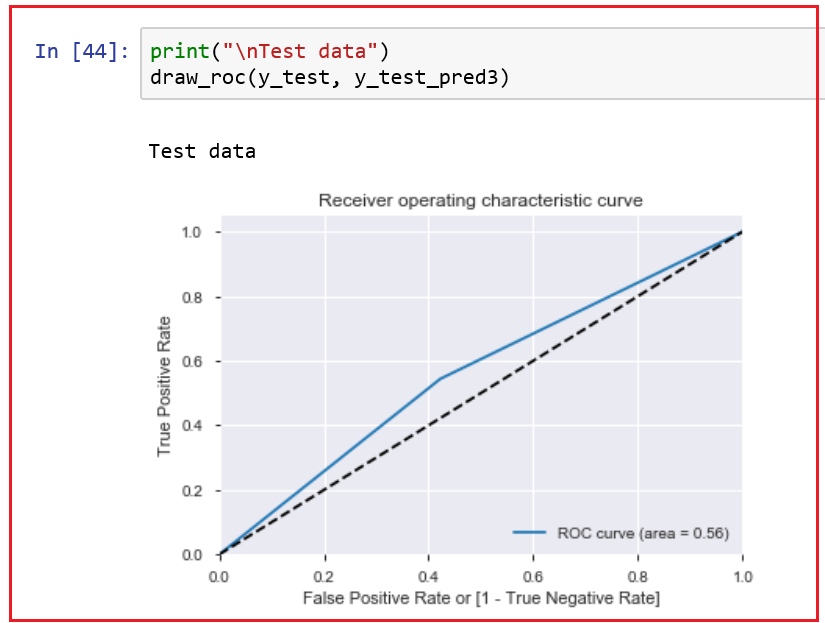
**

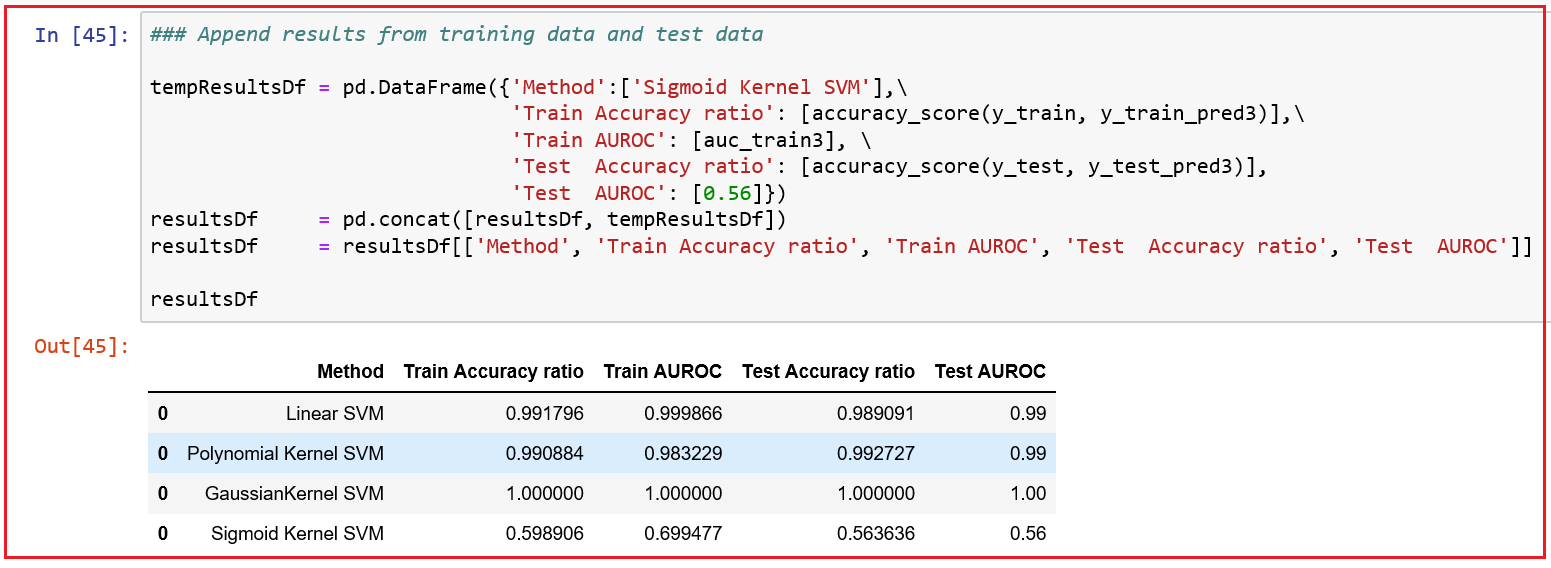
**

**

**

**

**



* ***We observe that the sigmoid kernel trick model did not perform well when compared to all the other three models we have seen so far. Number of cases mis-classified is 120 out of 275 observations when applied to our test data.***
* ***Guassian kernel is the best SVM model for our data.***

### SUPPORT VECTOR REGRESSION

* *Regression models predict continuous or real values based on a training dataset.*
* *Support Vector Regressor (SVR) is a Regression algorithm working with continuous values.*

**Terminology used:**

1. **Kernel** is the function that maps a lower dimensional data into a higher dimensional data.
2. **Hyper-plane** is a line that helps us to predict the continuous value or target value.
3. **Boundary line** creates a margin which can be on the boundary lines or outside it. They are the lines which are at a distance of ε (+ε and - ε) from the hyper-plane. Assume our hyper-plane is a straight line passing through the y-axis.

ax + b = 0 where a is the slope and b the intercept.

Two equations of the boundary lines are:

* Lower boundary line: ax + b = - ε
* Upper boundary line: ax + b = ε

Rewriting the above equation, - ε ≤ y - ax - b ≥ + ε given the fact that y = ax + b

1. **Support vectors** are the data points which are closest to the boundary. The distance of the points is the least.

**Difference between Simple regression model and Support Vector Regression**

* In Simple Regression, the goal is to minimize the error rate. In SVR, we fit the error rate within a certain threshold. In SVR, the best fit line is the line hyper-plane that has maximum number of points.
* In SVR, we try to decide a decision boundary at ε distance from the hyper-plane such that data points closest to the hyper-plane or the support vectors are within that boundary line.

**SVR model approximates the best values with a given margin called the ε-tube considering the model complexity and error rate.**

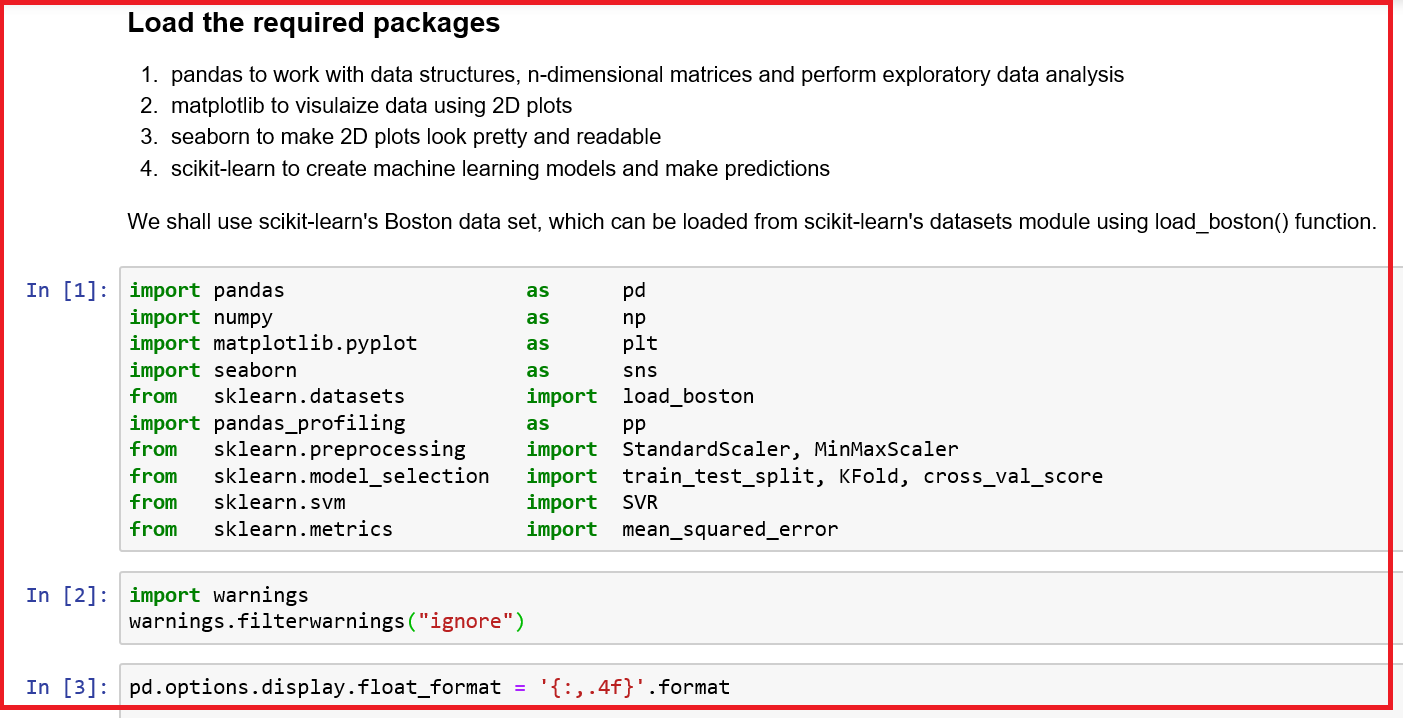
**Data**

* We will use Boston Housing price data set to solve a supervised learning regression problem. Besides location and square footage, a house value is determined by various other factors.

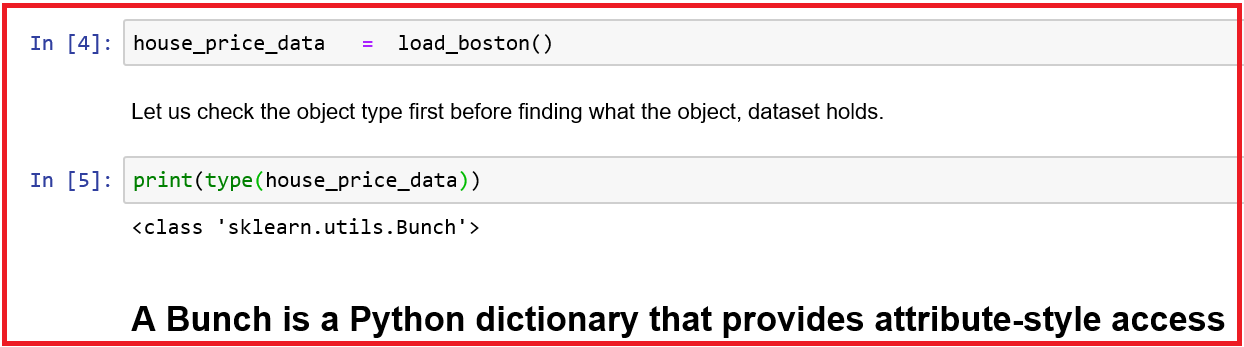
**Load the required packages**

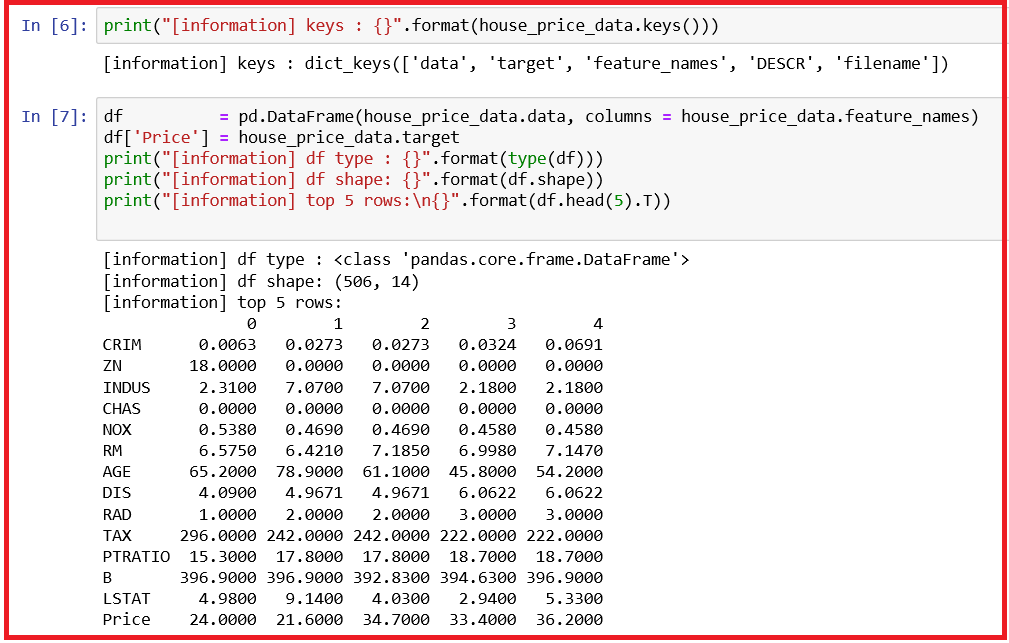
1. **pandas to work with data structures, n-dimensional matrices and perform exploratory data analysis**
2. **matplotlib to visulaize data using 2D plots**
3. **seaborn to make 2D plots look pretty and readable**
4. **scikit-learn to create machine learning models and make predictions**

* We shall use scikit-learn's Boston data set, which can be loaded from scikit-learn's datasets module using load\_boston() function.



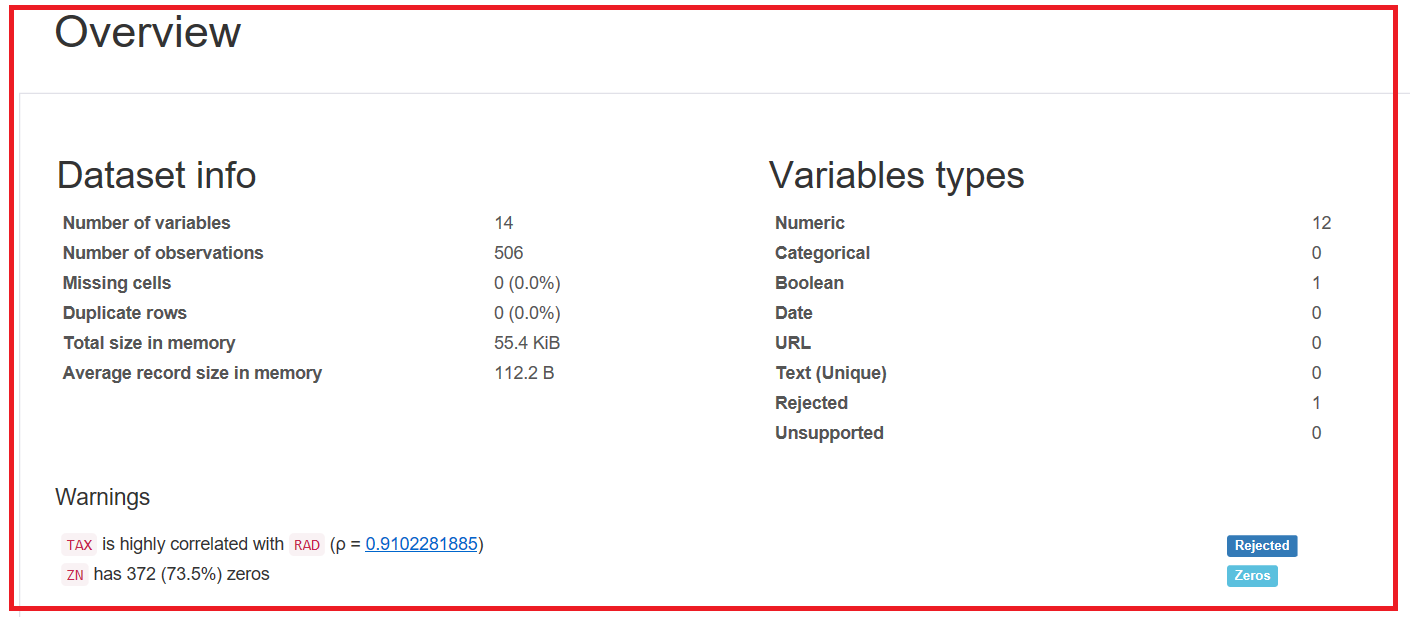
* Use scikit-learn to load the data set and pandas profiling to do quick Exploratory Data Analysis. The data set is loaded from scikit-learn's datasets module using load\_boston function.





* There are four keys in the data set, namely data, target, feature\_names and DESCR.
* We can access 13 features and 1 target using data key and target key respectively. Column names of the features or independent variables can be accessed using the key feature\_name. To get the description of each column, use DESCR key.

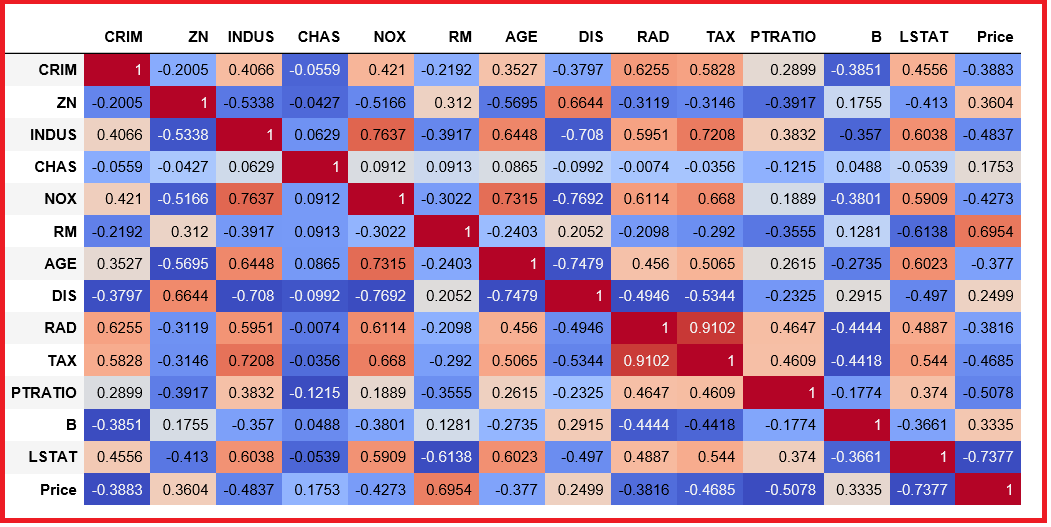
**Data overview**



**Correlations**

* To check the patterns in the data set, you need to use correlation between variables.
* Correlation coefficients fall between -1 and 1; 1 is perfect positive correlation, -1 is perfect negative correlation and 0 means no correlation.

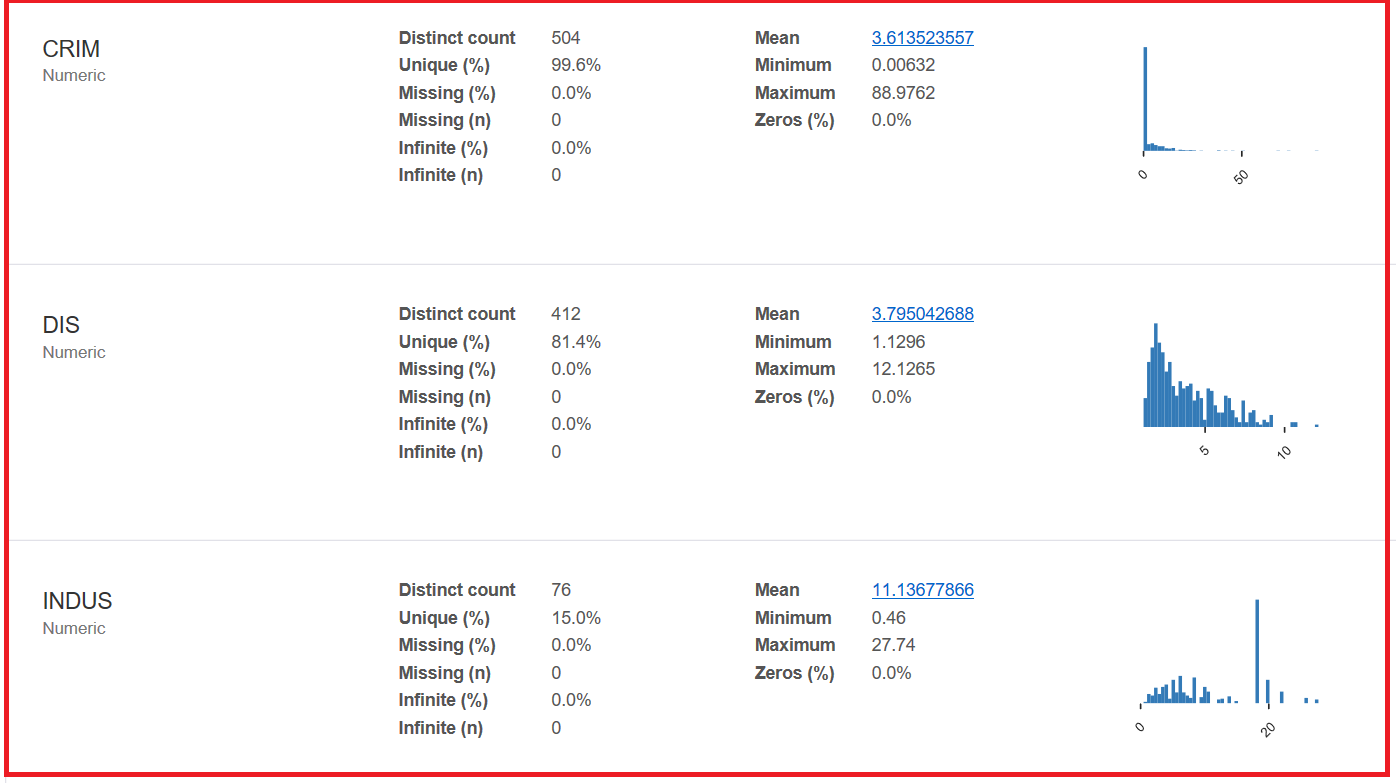


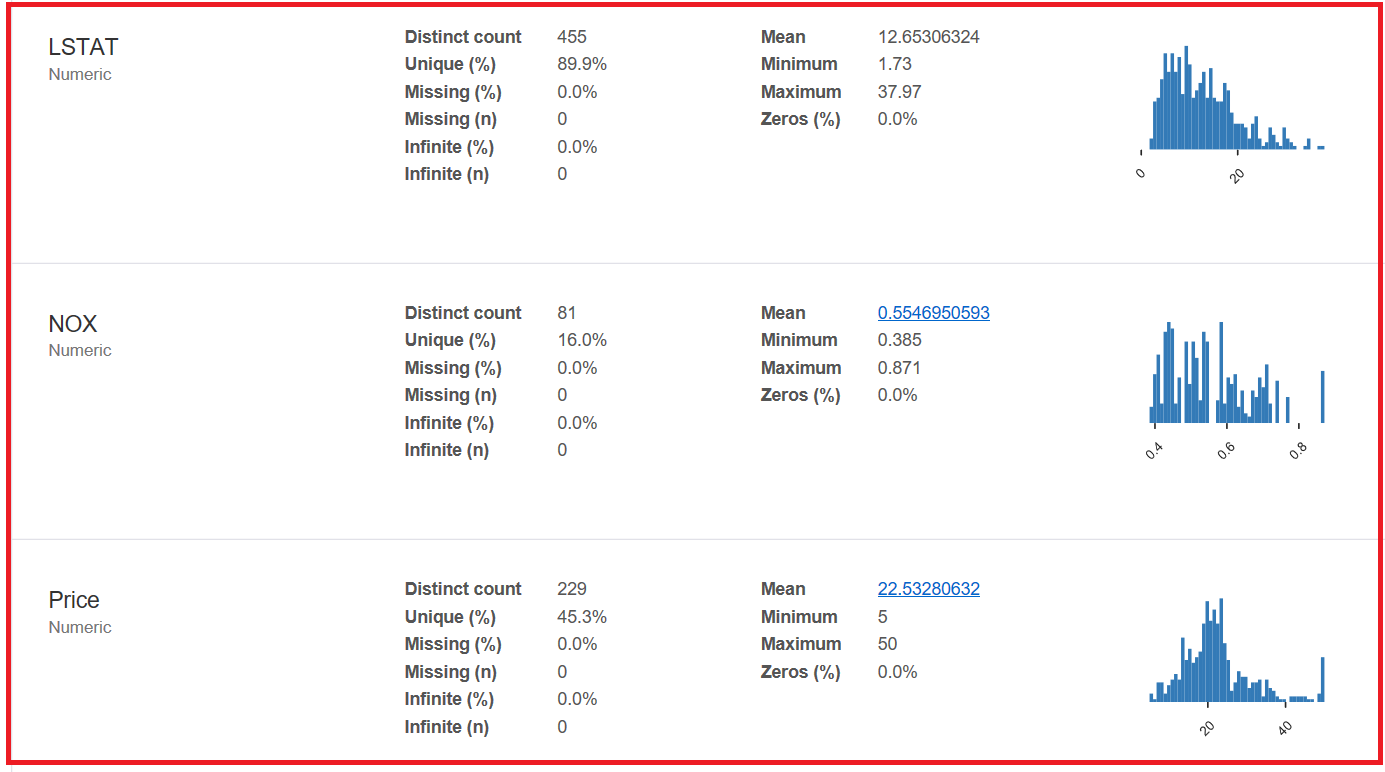


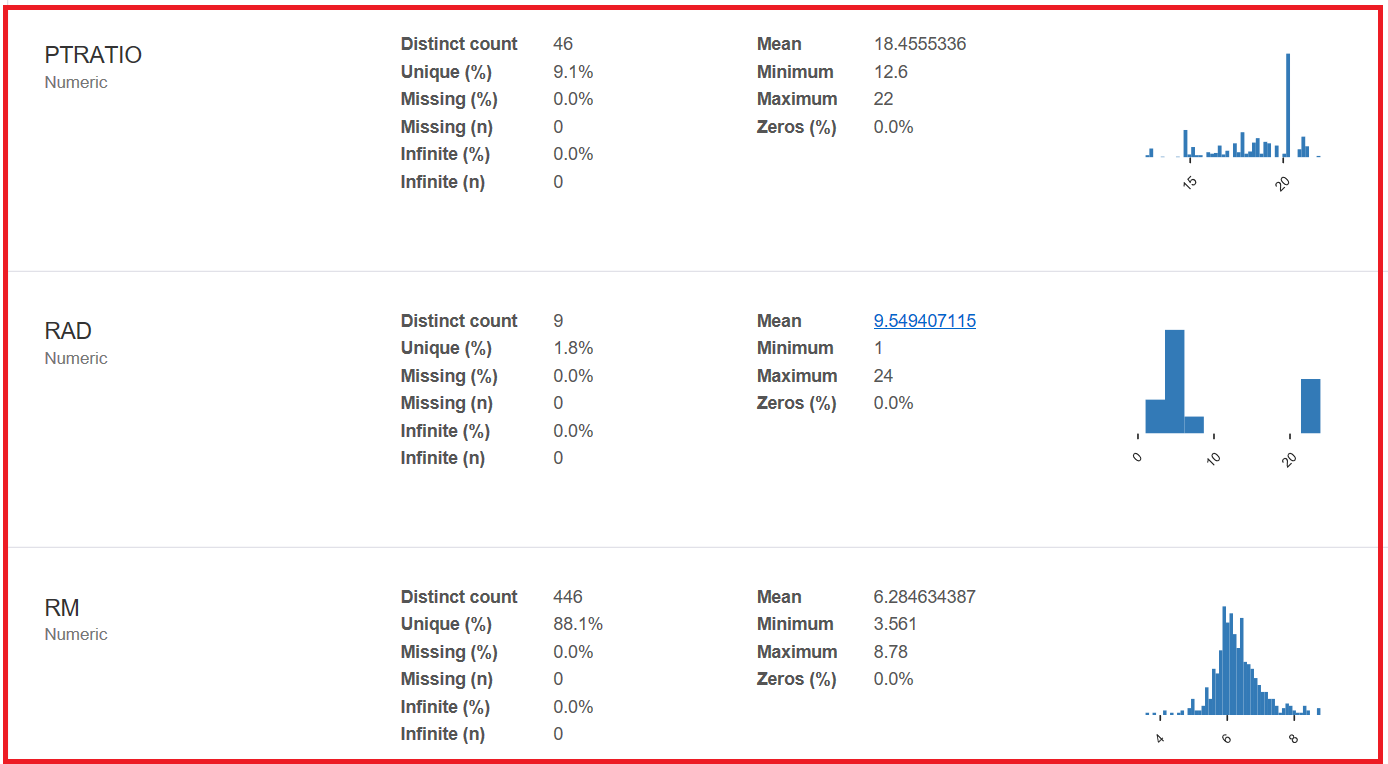
* We find very high correlation (0.9102) between Tax and RAD.

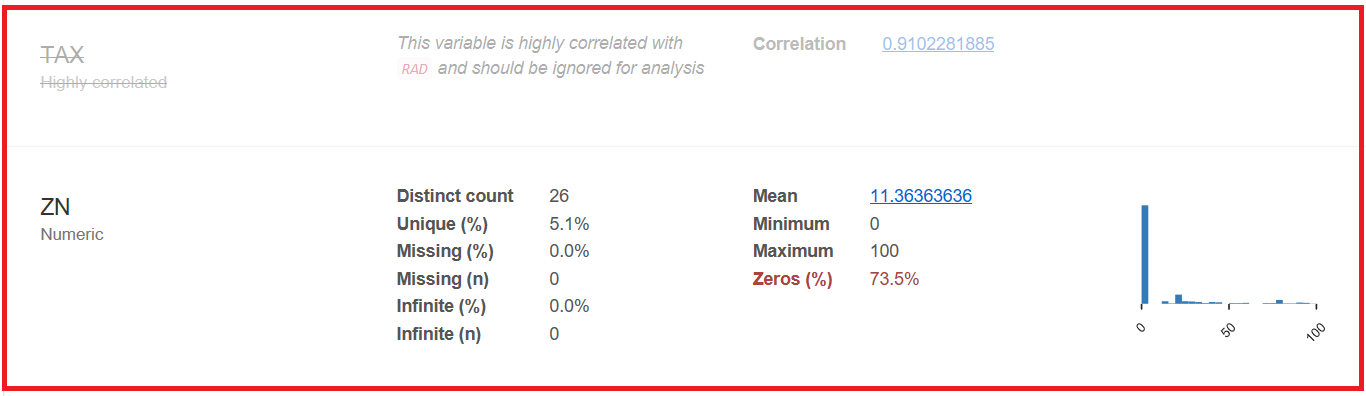
**Variables and their distribution**











* We observe from the histogram plots, CRIM, AGE, B and ZN have exponential distribution.
* NOX, RM and LSTAT is having a skewed gaussian distribution.
* RAD and TAX have bimodal distribution.

**Inference**

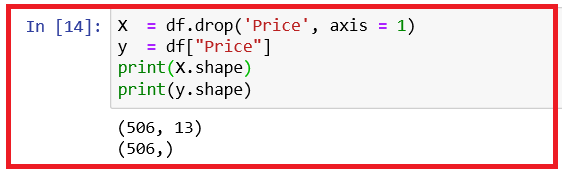
* We see a lot of structure in this dataset with outliers and different data distributions.
* From these visualizations, we infer that

1. Data is not standardized (meaning there are different data distributions).
2. Data is not normalized (meaning there are differing scales of data).

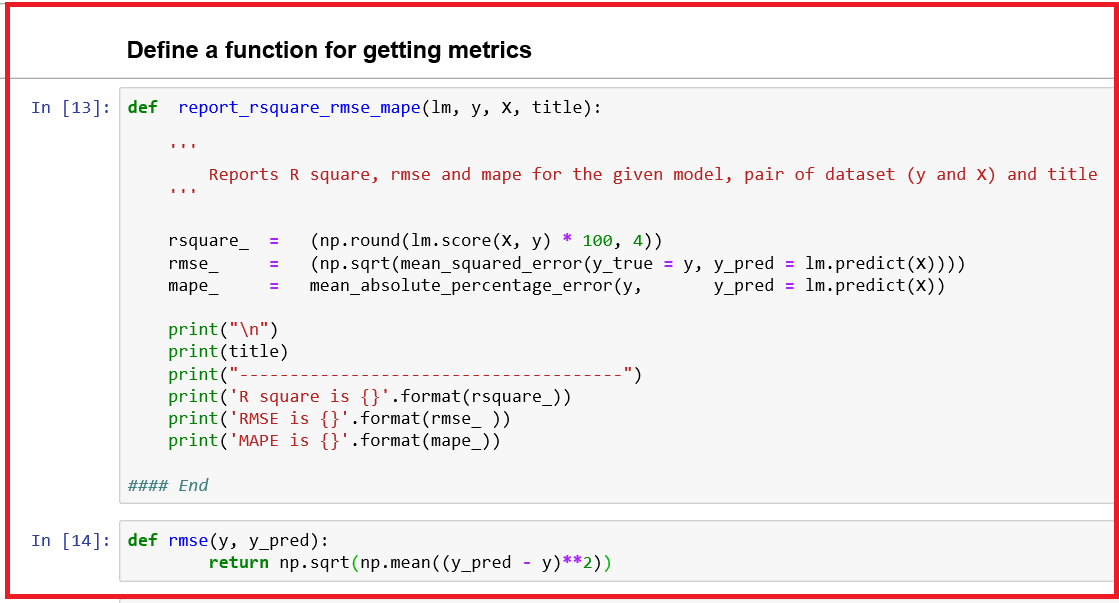
**Model building**

* We shall try SVR witha ten-fold cross validation method.

**Split the data into features and target**

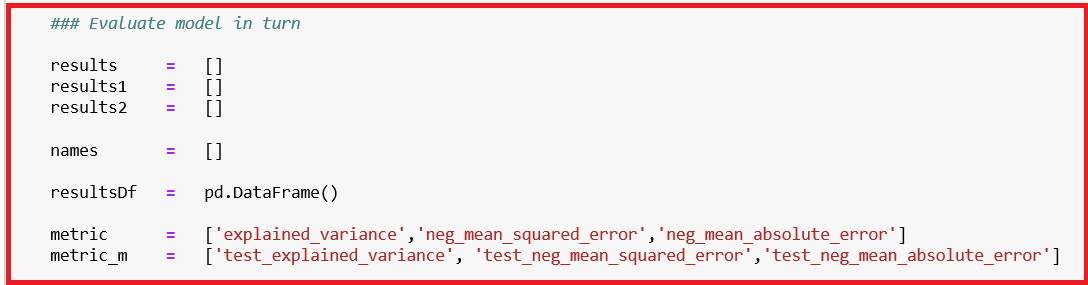


* Since we see different data distributions, we will standardize the dataset using StandardScaler function in scikit-learn. This is a useful technique where the attributes are transformed to a standard gaussian distribution with a mean of 0 and a standard deviation of 1.
* Define useful metrics such as R square which gives the explained variance in the target variable by the model, RMSE, root mean square error which is a is a frequently used measure of the differences between values (sample values) predicted by an estimator and the values observed and MAPE, Mean Absolute Percentage Error. The absolute value of the deviation between observed and predicted values is summed for every forecasted point and divided by the number of fitted points. Multiplying by 100 makes it a percentage error.

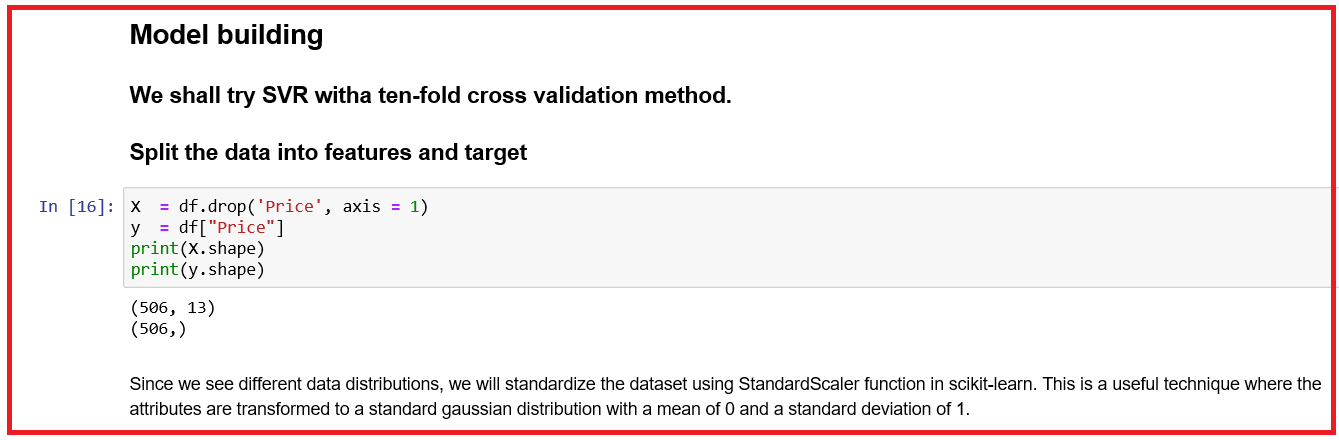


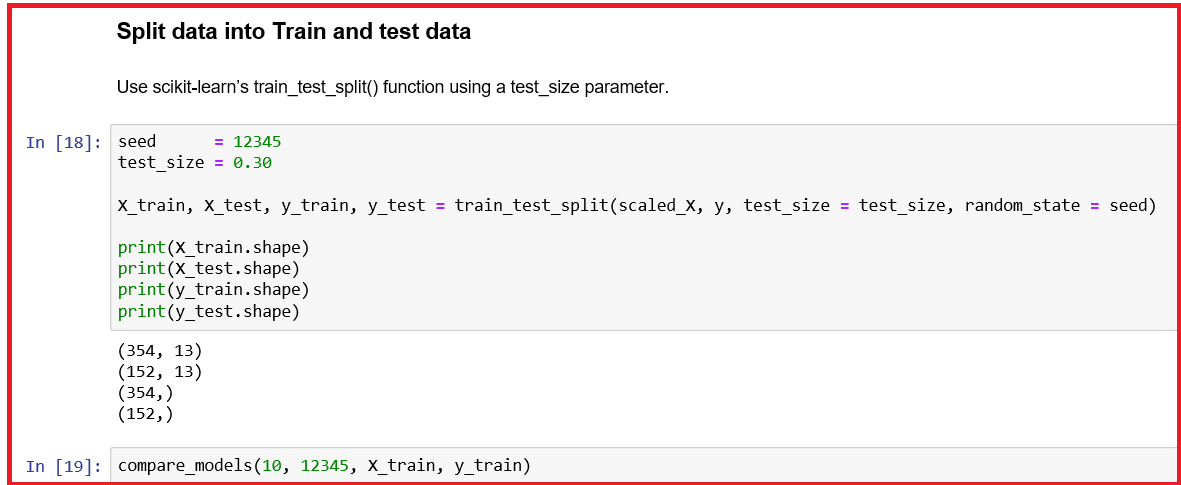
* We shall try different kernels such as rbf, linear, polynomial and sigmoid with different values for hyper-parameters, C = 1, 10, 100 and fixed value for epsilon = 0.01. For polynomial kernel, we shall use parameter, degree = 3 (default value).
* Here, kernel, degree, C, and epsilon are important parameters.
* kernel specifies the kernel type such as linear, poly, rbf and sigmoid
* degree is the degree for polynomial function and this is ignored for all other kernels
* C is the penalty parameter C of the error term
* epsilon specifies the epsilon-tube within which no penalty is associated in the training loss function with points predicted within a distance from the actual value.

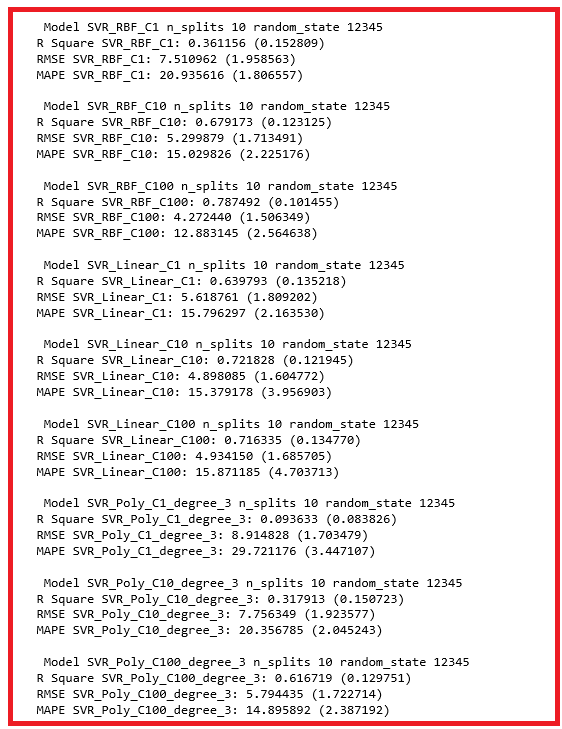


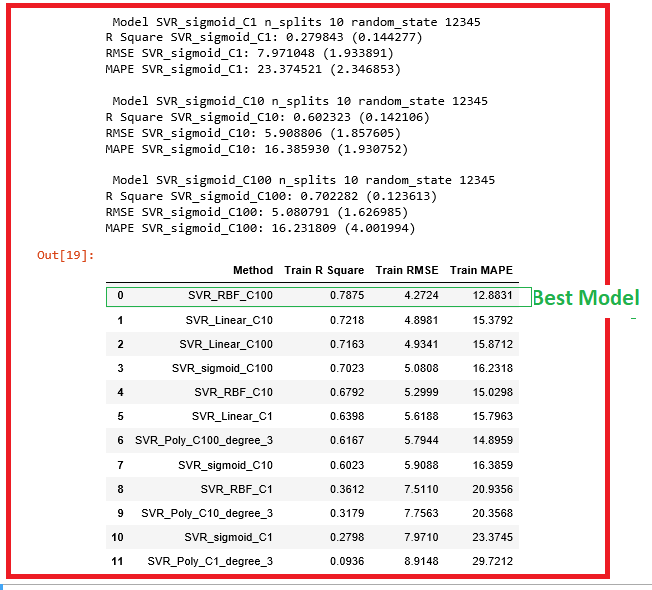


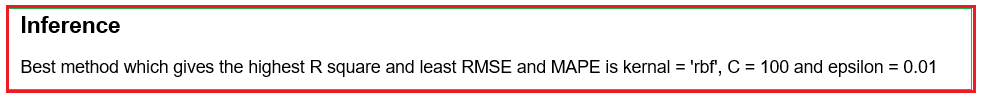




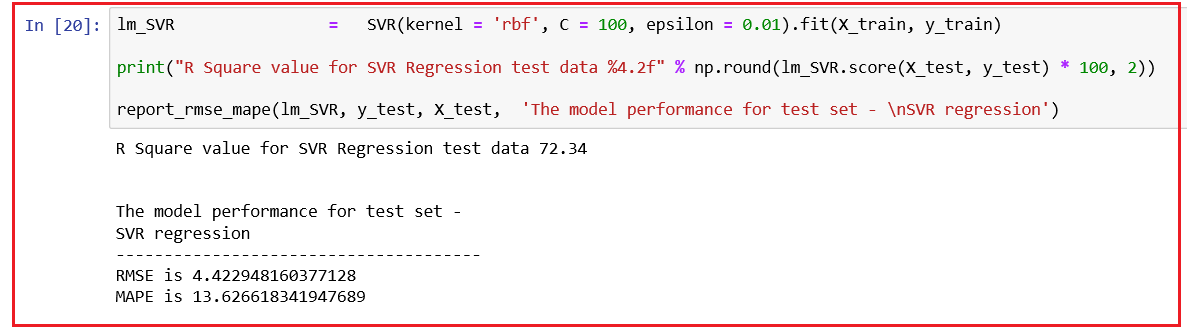








* We shall select this model and compare the performance on the test data.



* We observe that there is no big difference in the evaluation measures between training and test data as shown below and there is no visible model overfit;
* R square for training is 78.75, while for test data it is 72.34
* RMSE for training data is 4.2724, while for test data it is 4.4229
* MAPE for training data is 12.8831, while for test data it is 13.6266