**DIABETES PREDICTION USING DIFFERENT CLASSIFICATION ALGORITMS**

**Sruthi.S**

**2019506098**

**The work done by me can be accessed from here:** [**github link**](https://github.com/SummerProjectTEAM-10/Diabets-Prediction/tree/SRUTHI)

**DECRIPTION OF MY MODULE:**

* Model building using **LOGISTIC REGRESSION** and **SVM** algorithm.
* Hyper parameter tuning is done.
* I have also evaluated the model accuracy with the help of confusion matrix.

**STEPS INVOLVED IN MY MODULE:**

1. Import the necessary package and load the dataset.
2. Build a correlation matrix using heatmap.
3. Pre process the dataset.
4. Splitting the dataset into 70% training data and 30% testing data.
5. Building a Logistic Regression classifier and Support Vector Machine classifier.
6. Predict the target Outcome.
7. Evaluating the accuracy of the model.
8. Model tuning (hyper parameter tuning) to improve the accuracy of the model.

**KEYWORD:**

**SUPERVISED LEARNING:**

A supervised algorithm is shown the**“right answer”** for a set of sample data and finds a function that approximates the relationship between the inputs and outputs. Both input data and output is fed into the machine.

**EXPLANATION:**

**Step 1:**  In the first step import all the necessary packages.

**Step 2:** We will calculate the accuracy for both before and after preprocessing.

**Step 3:** In the next step we will be splitting the dataset into training data and testing data. It is an important step because we prevent our model from over fitting and to accurately evaluate our model.

* Training data – It is necessary to teach an ML algorithm.
* Testing data – It helps us to validate the progress of the algorithm’s training & optimize it for improved results.

There are different methods for splitting the dataset, the most common following the pareto ratio of 80:20 or sometimes 70:30.

**Step 4:**  Next is the model building phase where we use different classification algorithm. A ML model is built by learning and generalizing from the training data. Then we apply the acquired knowledge to the new data to make prediction. Detailed explanation about Logistic Regression and Support vector machine will be discussed next.

**Step 5:** Next is the prediction phase. After we have a model we will use that model to make predictions of target ,Y, for new data by plugging those new features to the model.

**Step 6:** To evaluate a classification model we can use accuracy, precision and recall. Accuracy is the percentage of correct predictions for the test data. It is calculated by dividing number of correct predictions by the number of total prediction.

**Step 7:** Fine tuning hyper parameters for optimal performance. Hyper parameters cannot be directly learned from the regular training process begins.

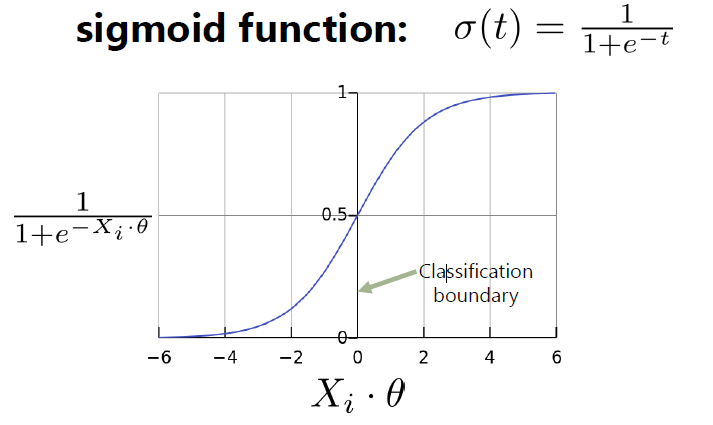
Eg: penalty in logistic regression classifier; C and sigma hyper parameters for SVM.

Two strategies used to perform hyper parameter tuning are:

* GridSearchCV
* RandomizedSearchCV
* **LOGISTIC REGRESSION**

Logistic regression is basically a supervised classification algorithm to find the probability of dependent variable. In a classification problem, the target variable(or output), y, can take only discrete values for given set of features(or inputs), X. Logistic regression becomes a classification technique only when a decision threshold is brought into the picture. The setting of the threshold value is a very important aspect of Logistic regression and is dependent on the classification problem itself.Logistic Regression uses functions called the logit functions that help drive a relationship between the dependent and independent variables by predicting the probabilities.

Logistic functions/sigmoid functions convert the probabilities into binary values which can be further used for predictions. It’s an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.



[IMAGE SOURCE](https://editor.analyticsvidhya.com/uploads/23302main-qimg-7fc9e8601c15e33945720800aa237a7f.png)

The decision for the value of the threshold value is majorly affected by the values of precision and recall. In applications where we want to reduce the number of false negatives without necessarily reducing the number false positives, we choose a decision value which has a low value of Precision or high value of Recall. For diabetes prediction application, we do not want any affected patient to be classified as not affected without giving much heed to if the patient is being wrongfully diagnosed with diabetes. This is because, the absence of diabetes can be detected by further medical diseases but the presence of the disease cannot be detected in an already rejected candidate.

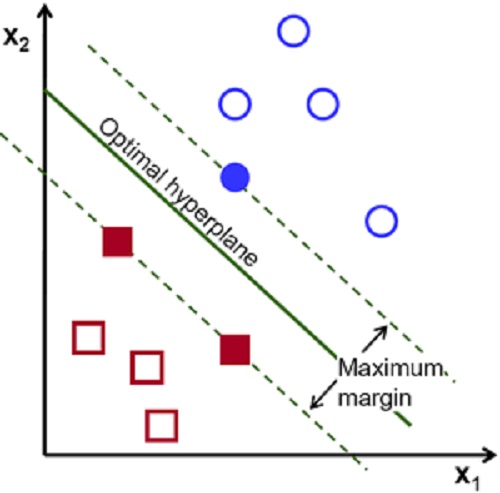
This algorithm can be implemented in two ways. The first way is to write your own functions i.e. you code your own sigmoid function, cost function, gradient function, etc. instead of using some library. The second way is, to use the Scikit-Learn library.

* **SUPPORT VECTOR MACHINE**

It is one of the supervised machine learning algorithm which is used to solve both classification and regression problem. We will perform classification by finding the hyper-plane that differentiates the class very well. Most commonly used supervised classification algorithms like logistic regression learn the differences in characteristics, but SVM learns the similarities between the characteristics.

**Hyper-plane** is a decision boundary which is constructed such that the distance between data points closest to hyper-plane in each class is maximum.

SVM creates 2 planes along with the hyper-plane, which passes through nearest positive points and nearest negative points. We choose the hyper-plane such that the marginal distance between them is maximum so that it provides more generalized model. SVM selects the hyper-plane which classifies the classes accurately prior to maximizing the margin.

[IMAGE SOURCE 1](https://www.aitrends.com/wp-content/uploads/2018/01/1-19SVM-2.jpg) [IMAGE SOURCE 2](https://static.javatpoint.com/tutorial/machine-learning/images/support-vector-machine-algorithm.png)

**Support vectors:** SVM algorithm finds the closest point of lines from both the classes. These points are called support vector and they pass through the marginal plane. It helps us to determine the maximum distance of the marginal plane. The hyper-plane with maximum margin is called optimal hyper-plane.

Non linear classification using SVM is done by mathematical function called kernel which converts lower dimension into higher dimension.

**HYPERPARAMETER TUNING:**

Model parameters are the ones that are an internal part of the model and their value is computed automatically by the model referring to the data like support vectors in a support vector machine. But hyper parameters are the ones that can be manipulated by the programmer to improve the performance of the model

Hyper parameter tuning is choosing a set of optimal hyper parameter for learning algorithm. Hyper parameters cannot be learned directly from the training process. It is a model argument whose value is set before learning process begins. It controls over-fitting and under-fitting of the model. Optimal hyper parameters often differ according to the dataset. Not all model hyper parameters are equally important. It is desirable to select a minimum subset of model hyper parameters to search because more hyper parameters of an algorithm that you need to tune, slower the training process. Different models have many hyper parameters, for finding the best combination of parameters, we have 2 methods namely:

* GridSearchCV
* RandomizedSearchCV

In GridSearchCV approach, machine learning model is evaluated for a range of hyper parameter values. This approach is called GridSearchCV, because it searches for best set of hyper parameters from a grid of hyper parameters values.

Drawback : GridSearchCV will go through all the intermediate combinations of hyper parameters which makes grid search computationally very expensive.  
   
**RandomizedSearchCV**  
RandomizedSearchCV solves the drawbacks of GridSearchCV, as it goes through only a fixed number of hyper parameter settings. It moves within the grid in random fashion to find the best set hyper parameters. This approach reduces unnecessary computation.

**LOGISTIC REGRESSON – HYPER PARAMETERS:**

* **Solver : *{‘newton-cg’, ‘lbfgs’, ‘liblinear’, ‘sag’, ‘saga’}, default=’lbfgs’***

Algorithm to use in the optimization problem.

* **Penalty: *{‘l1’, ‘l2’, ‘elasticnet’, ‘none’}, default=’l2’***

Specify the norm of the penalty

* **C *float, default=1.0***

Inverse of regularization strength; must be a positive float.

**SVM – HYPER PARAMETERS:**

* **C *float, default=1.0***

Regularization parameter. Strength of the regularization is inversely proportional to C.

* **kernel: *{‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’}, default=’rbf’***

It specifies the kernel type to be used in the algorithm.

* **Gamma: *{‘scale’, ‘auto’} or float, default=’scale’***

Kernel coefficient for ‘rbf’, ‘poly’ and ‘sigmoid’.

**DT – HYPER PARAMETERS:**

* **criterion: *{“gini”, “entropy”}, default=”gini”***

The function to measure the quality of a split.

* **Splitter: *{“best”, “random”}, default=”best”***

The strategy used to choose the split at each node.

* **max\_depth: *int, default=None***

The maximum depth of the tree. If None, then nodes are expanded until all leaves contain less than min\_samples\_split samples.

* **min\_samples\_split: *int or float, default=2***

The minimum number of samples required to split an internal node

* **min\_samples\_leaf: *int or float, default=1***

The minimum number of samples required to be at a leaf node.

* **max\_features : *int, float or {“auto”, “sqrt”, “log2”}, default=None***

The number of features to consider when looking for the best split

**NB – HYPER PARAMETERS:**

* **var\_smoothing : *float, default=1e-9***

Portion of the largest variance of all features that is added to variances for calculation stability.

**RF – HYPER PARAMETERS:**

* **n\_estimators : *int, default=100***

The number of trees in the forest.

* **criterion: *{“gini”, “entropy”}, default=”gini”***

The function to measure the quality of a split.

* **max\_depth: *int, default=None***

The maximum depth of the tree. If None, then nodes are expanded until all leaves contain less than min\_samples\_split samples.

* **min\_samples\_split: *int or float, default=2***

The minimum number of samples required to split an internal node

* **min\_samples\_leaf: *int or float, default=1***

The minimum number of samples required to be at a leaf node.

* **max\_features : *int, float or {“auto”, “sqrt”, “log2”}, default=None***

The number of features to consider when looking for the best split

**KNN – HYPER PARAMETERS:**

* **n\_neighbors : *int, default=5***

Number of neighbors to use by default for kneighbors queries.

**ALGORITHMS:**

**INPUT:** Dataset values

**OUTPUT:** Prediction is done and accuracy is returned.

**PROCEDURE:**

Read the dataset from the user.

Pre-process the dataset and perform feature engineering.

Select the optimal values for hyper parameters.

While (stop condition is not met) do

Build the model using SVM and LR

Implement the SVM and LR train step for each data point

Implement SVM and LR classify for testing data points

Calculate accuracy for test data and train data.

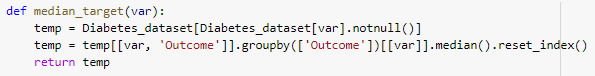
end while

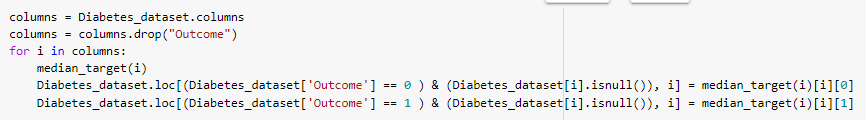
Return accuracy

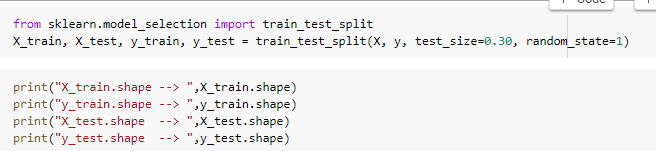
**DATA STUCTURES USED:**

Numpy array and dataframe in pandas.

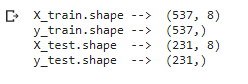
**PRE PROCESSING THE DATA:**

3.1.PNGReplacing zero values with NULL and the replacing it by median.



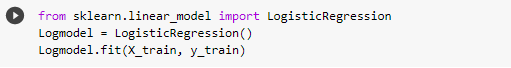
**SPLITTING THE DATASET**

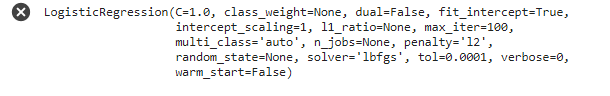
**CODE:**



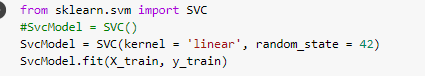
**OUTPUT SCREENSHOT**

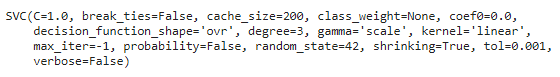
**BUILDING LOGISTIC REGRESSION CLASSIFIER**

**CODE:**

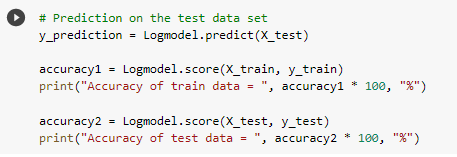
**OUTPUT SCREENSHOT:**

**BUILDING SUPPORT VECTOR MACHINE CLASSIFIER**

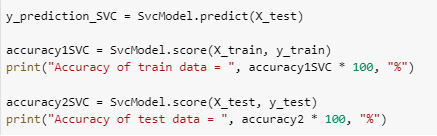
**CODE:**

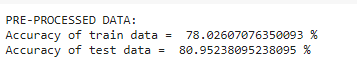
**OUTPUT SCREENSHOT:**

**PREDICTING THE TARGET OUTCOME**

**LOGISTIC REGRESSION CODE:**

**6.3.PNG6.2.PNGOUTPUT SCREENSHOT: (WITHOUT & WITH PRE-PROCESSING)**

**SVM CODE:**

**6.5.PNGOUTPUT SCREENSHOT: (WITHOUT & WITH PRE-PROCESSING)**

**AFTER HYPER PARAMETER TUNING – ACCURACY:**

