Machine Learning Engineer Nanodegree

Capstone Proposal
Anuchuri Teja
April 24, 2019

Capstone Project
April 25, 2019

Diabetes

I. Definition

Project Overview:

- India is regarded as the “Diabetes Capital” of the world owing to the existence of the largest number of people with diabetes in this country. Diabetes is a serious public health problem that has a strong negative impact on the health-related quality of life (HRQoL).
- The dataset created is a part of research in which the researchers are trying to introduce a new parameterizable metric that measures the whether or not each patient will have an onset of diabetes.
- The diabetes dataset contains some attributes like Pregnancies,Glucose,BloodPressure,SkinThickness,Insulin,DiabetesPedigreeFunction, BMI, Age was used to predict diabetes or not where predict outcome is either 0 or 1.
- For this dataset there is no need to data preprocessing because We can’t find any missing or null data points of the data set and then you know The data already preprocessed.
- We can choose f1_score as a metric because our dataset is not balance in outcome
  0 ⇒ 1316
  1 ⇒ 684
- This dataset may be used for a variety of tasks, the most obvious of which is diabetes via classification. Using classification algorithms like Logistic Regression, Decision Trees, Random forests etc.
- We find f1_scores of all the models, that we applied on the data gridsearchcv algorithm on best classification model.
- [1] Liping Wei and Russ B. Altman. An Automated System for Generating Comparative Disease Profiles and Making Diagnoses . Section on Medical Informatics Stanford University School of Medicine, MSOB X215. [View Context].

Problem Statement

- The dataset i am considering is from https://www.kaggle.com/johndasilva/diabetes
- This dataset contains diabetes data with nine different attributes.
- Inputs are attributes Pregnancies,Glucose,BloodPressure, SkinThickness,Insulin,DiabetesPedigreeFunction, BMI,Age and then ouput will be outcome is to look at the diabetes dataset and predict whether diabetes or not.
- The tasks are:
  1. Download the dataset
  2. Visualizing the data
  3. Split the data and train and test which classifier perform well on the dataset based on the evaluation metric we have chosen.
  5. Choose the best classifier that gives best f1_score.
  6. Apply gridsearchcv to optimal best classification model f1_score.
  7. To find cross_valid_score whether the find best classification model is robust or not.

Metric:

- After observing the dataset outcomes classes are inbalanced. We can choose a simple metric like f1_score to calculate the performance of our algorithm.The formula for the F1 score is:

```
F1 = 2 * (precision * recall) / (precision + recall)
```

- The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal.

II. Analysis

Data Exploration:

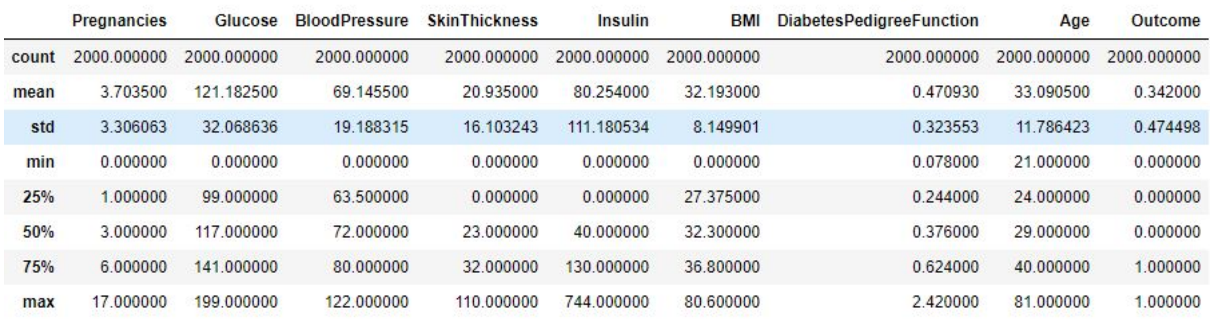
- Data is provided as a CSV file.
- Note that the last column from this dataset, `'outcome'`, will be our target label.
- All other columns are features about each individual in the diabetes database.

The dataset is contains 2000 instances, 9 attributes.

The attributes are:

1. Pregnancies	: Number of times pregnant
2. Glucose	: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. BloodPressure	: Diastolic blood pressure (mm Hg)
4. SkinThickness	: Triceps skin fold thickness (mm)
5. Insulin	: 2-Hour serum insulin (mu U/ml)
6. BMI	: Body mass index (weight in kg/(height in m)^2)
7. DiabetesPedigreeFunction	: Diabetes pedigree function
8. Age	: Age (years)
9. Outcome	: Class variable (1:tested positive for diabetes, 0: tested negative for diabetes)

- The shape of the dataset is **(2000, 9).**
- In the diabetes dataset total 9 attributes columns.
- Statistics analysis



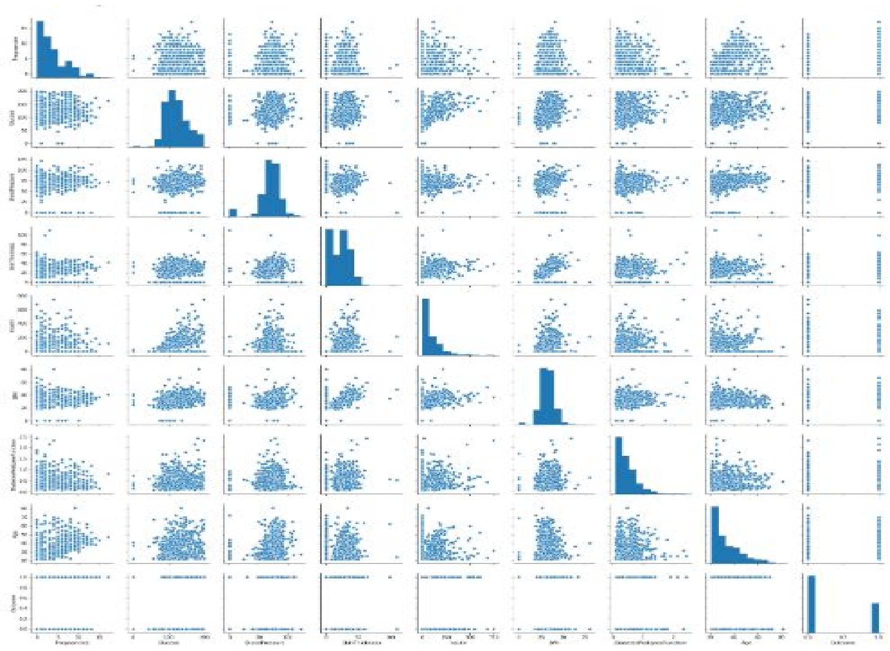
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
mean	3.703500	121.182500	69.145500	20.935000	80.254000	32.193000	0.470930	33.090500	0.342000
std	3.306063	32.068636	19.188315	16.103243	111.180534	8.149901	0.323553	11.786423	0.474498
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	63.500000	0.000000	0.000000	27.375000	0.244000	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	40.000000	32.300000	0.376000	29.000000	0.000000
75%	6.000000	141.000000	80.000000	32.000000	130.000000	36.800000	0.624000	40.000000	1.000000
max	17.000000	199.000000	122.000000	110.000000	744.000000	80.600000	2.420000	81.000000	1.000000

Exploratory Visualization:

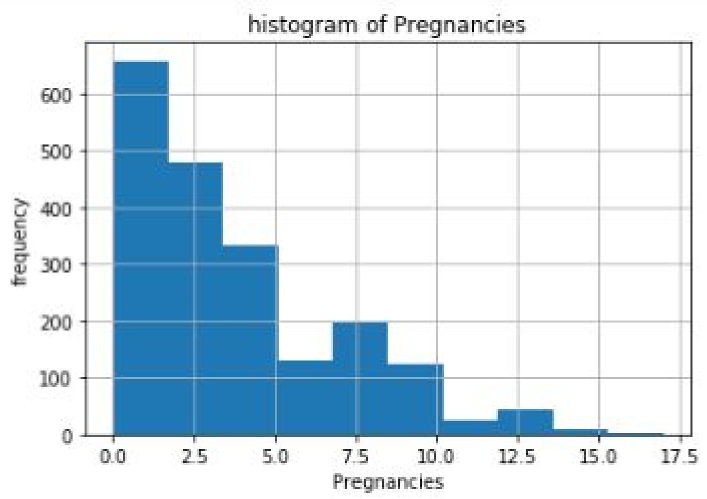
- we can not extract important features because every feature has its own importance in finding diabetes or not.
- We can find any missing or null data points of the data set (if there is any) using the following pandas function.

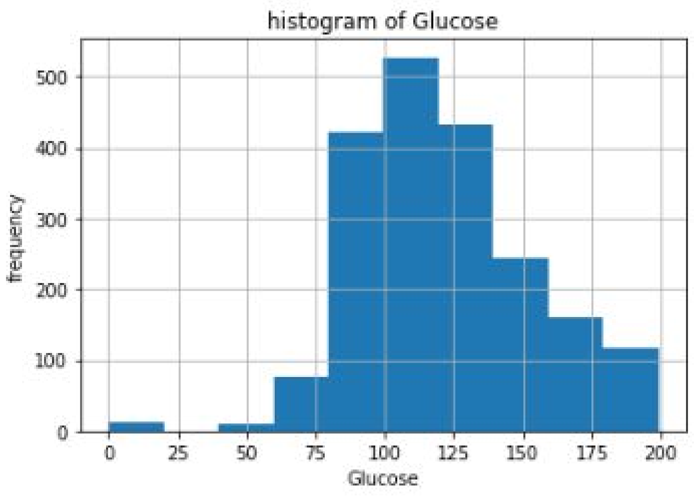
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

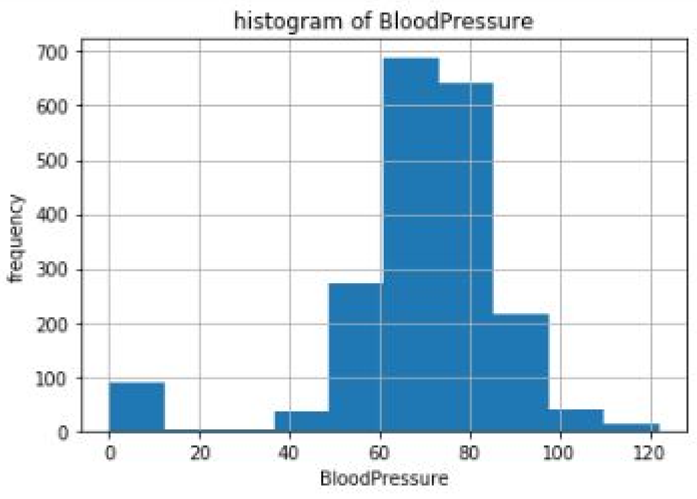
- Fortunately, for this dataset, there are no invalid or missing entries we must deal with, however, there are some qualities about certain features that must be adjusted. This preprocessing can help tremendously with the outcome and predictive power of nearly all learning algorithms.
- We can observe that there are no data points missing in the data set. If there were any, we should deal with them accordingly.

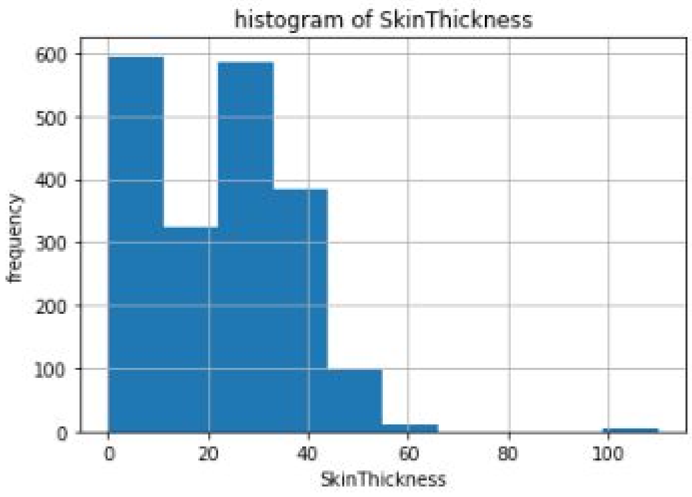


- The data is correlation, we can observe correlation distribution of data above
- Each and very columns attribute histogram(x-axis= “column attribute”,y-axis= “frequency”)

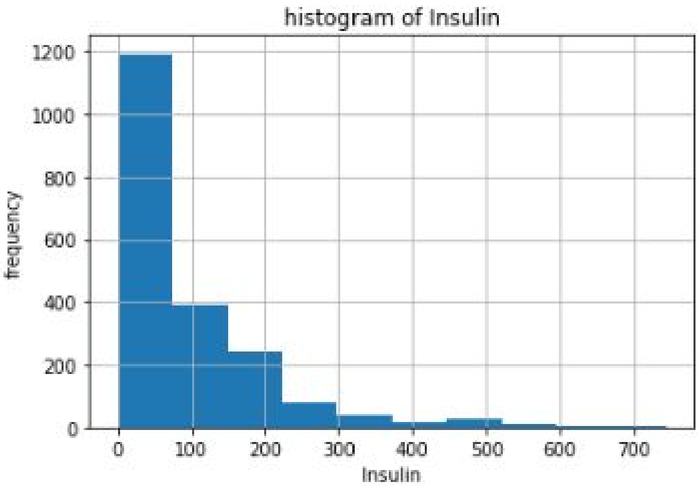

histogram of Pregnancies
frequency
Pregnancies
0
100
200
300
400
500
600
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


histogram of Glucose
frequency
Glucose
0
100
200
300
400
500
0
25
50
75
100
125
150
175
200

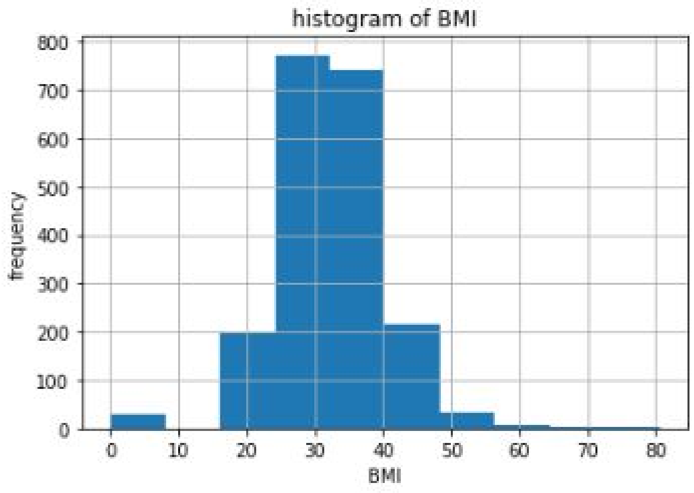

histogram of BloodPressure
frequency
BloodPressure
0
100
200
300
400
500
600
700
0
20
40
60
80
100
120



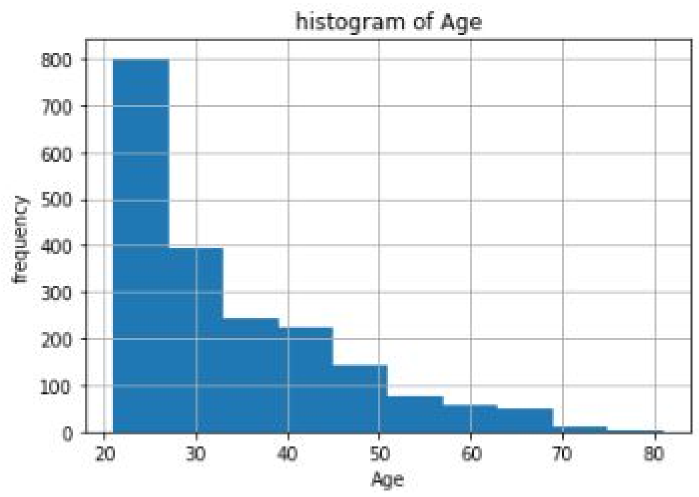
histogram of SkinThickness
frequency
0
100
200
300
400
500
600
0
20
40
60
80
100
SkinThickness

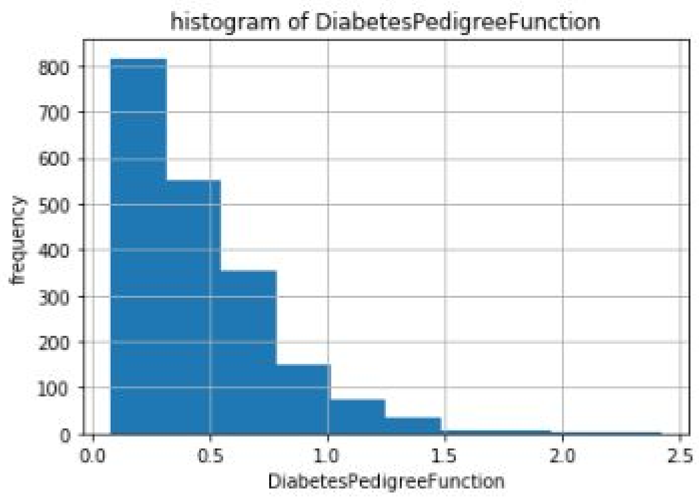


histogram of Insulin
frequency
0
200
400
600
800
1000
1200
0
100
200
300
400
500
600
700
Insulin

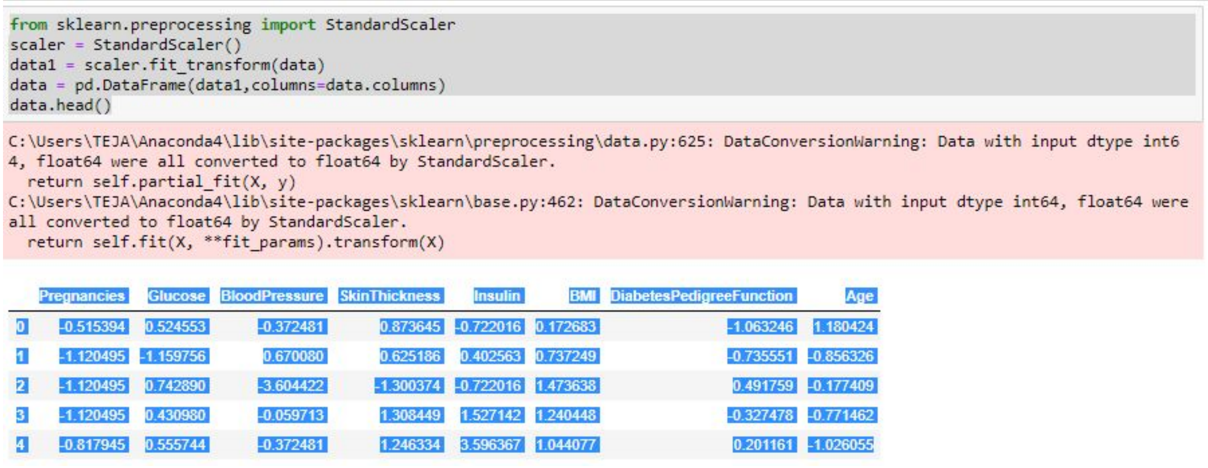


histogram of BMI
frequency
0
100
200
300
400
500
600
700
800
0
10
20
30
40
50
60
70
80
BMI

●




- Intitaly features of data is not within a particular range.so we need to use scaling hear.
- It is a step of Data Pre Processing which is applied to independent variables or features of data. It basically helps to normalise the data within a particular range. Sometimes, it also helps in speeding up the calculations in an algorithm.



```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
data1 = scaler.fit_transform(data)
data = pd.DataFrame(data1,columns=data.columns)
data.head()
```

```
C:\Users\TEJA\Anaconda4\lib\site-packages\sklearn\preprocessing\data.py:625: DataConversionWarning: Data with input dtype int6
4, float64 were all converted to float64 by StandardScaler.
  return self.partial_fit(X, y)
C:\Users\TEJA\Anaconda4\lib\site-packages\sklearn\base.py:462: DataConversionWarning: Data with input dtype int64, float64 were
all converted to float64 by StandardScaler.
  return self.fit(X, **fit_params).transform(X)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	-0.515394	0.524553	-0.372481	0.873645	-0.722016	0.172683	-1.063246	1.180424
1	-1.120495	-1.159756	0.670080	0.625186	0.402563	0.737249	-0.735551	-0.856326
2	-1.120495	0.742890	-3.604422	-1.300374	-0.722016	1.473638	0.491759	-0.177409
3	-1.120495	0.430980	-0.059713	1.308449	1.527142	1.240448	-0.327478	-0.771462
4	-0.817945	0.555744	-0.372481	1.246334	3.596367	1.044077	0.201161	-1.026055

●

Algorithms and Techniques:

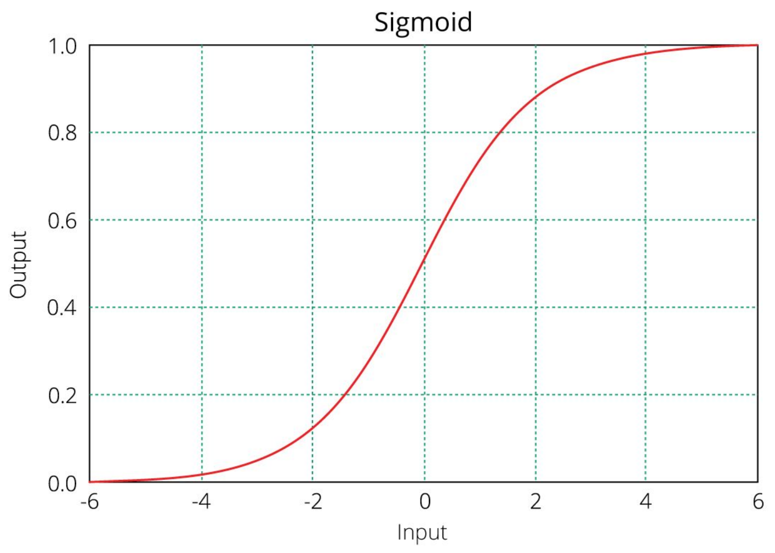
1. Logistic Regression:

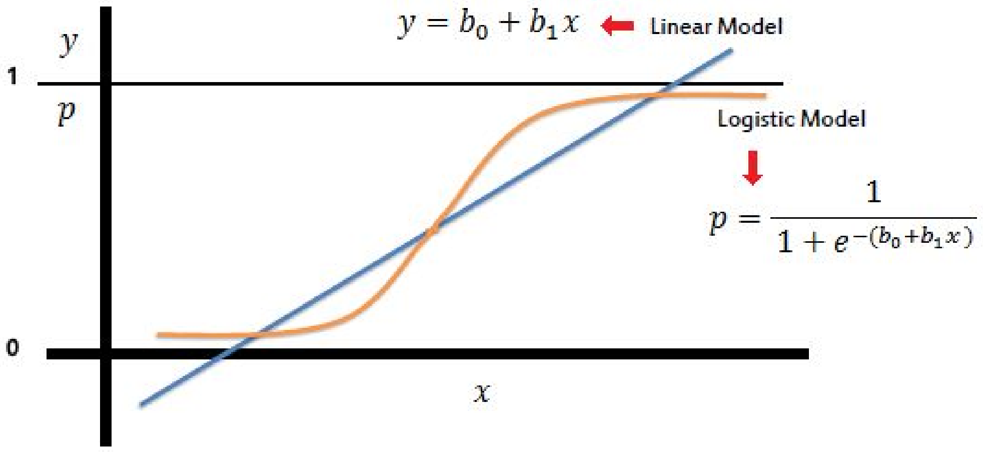
- Logistic Regression is a classification algorithm. It is used to predict a binary outcome (1 / 0, Yes / No, True / False) given a set of independent variables.

- To represent the binary/categorical outcome, we use dummy variables. You can also think of logistic regression as a special case of linear regression when the outcome variable is categorical, where we are using a log of odds as the dependent variable. In simple words, it predicts the probability of occurrence of an event by fitting data to a logit function.

- Logistic Regression belongs to the family of generalized linear models. It is a binary classification algorithm used when the response variable is dichotomous (1 or 0). Inherently, it returns the set of probabilities of the target class. But, we can also obtain response labels using a probability threshold value. Following are the assumptions made by Logistic Regression:

    1. The response variable must follow a binomial distribution.
    2. Logistic Regression assumes a linear relationship between the independent variables and the link function (logit).
    3. The dependent variable should have mutually exclusive and exhaustive categories.

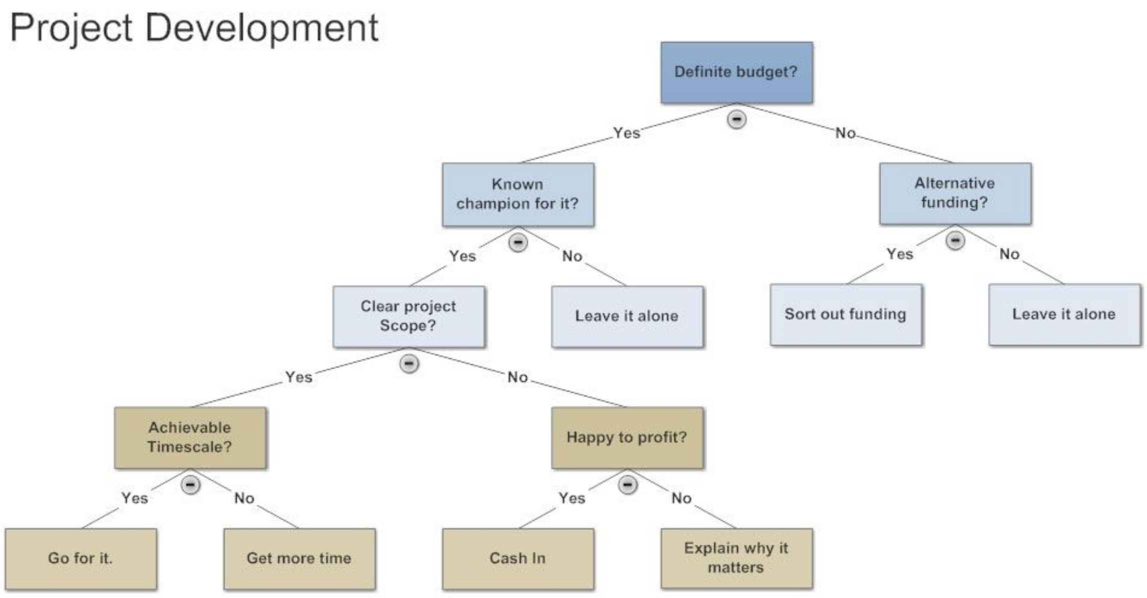




2. Decision Trees (Supervised Learning – Classification/Regression):

- A decision tree is a flow-chart-like tree structure that uses a branching method to illustrate every possible outcome of a decision.

- Each node within the tree represents a test on a specific variable – and each branch is the outcome of that test.



Real time example:

Direct Marketing, Fraud Detection

- **Strengths**:
  It is very easy to understand and interpret. The data for decision trees require minimal preparation.

- **Weaknesses**:
  - Sometimes a decision tree may become complex. The outcomes of decisions can be based mainly on your expectations.

  - So this can lead to unrealistic decision trees. Since a decision tree can handle both numerical and categorical data, it's a good choice of algorithm.

  - The goal is to create a model that predicts the value of the target variable by learning simple decision rules.

  - Decision trees tend to have high variance when they utilize different training and test sets of the same data since they tend to overfit on training data. This leads to poor performance on unseen data. Unfortunately, this limits the usage of decision trees in predictive modelling.

  - However, using ensemble methods, we can create models that utilize underlying decision trees as a foundation for producing powerful results.

  References:
  - Decision Trees link-1
  - Decision Trees link-2

3. SVM classifier:

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection.

Real time example:

Protein fold and remote homology detection,Face Detection

Strengths:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

Weaknesses:

- If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation (see Scores and probabilities, below).

References:

- **SVM link-1**
- **SVM link-2**

4. RandomForestClassifier (Supervised Learning – Classification):

- Random forests or ‘random decision forests’ is an ensemble learning method, combining multiple algorithms to generate better results for classification, regression and other tasks.
- Each individual classifier is weak, but when combined with others, can produce excellent results.

- The algorithm starts with a ‘decision tree’ (a tree-like graph or model of decisions) and input is entered at the top. It then travels down the tree, with data being segmented into smaller and smaller sets, based on specific variables.

Real Time Example:

- Random forest model can be applied in the medical domain to identify a disease based on symptoms. Example: detection of Alzheimer's disease.

Strengths and weaknesses:

- Random forest runtimes are quite fast, and they are able to deal with unbalanced and missing data. Random Forest weaknesses are that when used for regression they cannot predict beyond the range in the training data and that they may overfit data sets that are particularly noisy.

Parameters:
https://scikitlearn.org/stable/modules/generated/sklearn.ensemble.RandomForestCl assifier.html

References:

- Random forest link-1

5. GridSearchCV

- GridSearchCV implements a “fit” and a “score” method.
- It also implements “predict”, “predict_proba”, “decision_function”, “transform” and “inverse_transform” if they are implemented in the estimator used.
- GridSearchCV parameters are estimator,param_grid,refift,n_job,cv,scoring..etc.
- The parameters of the estimator used to apply these methods are optimized by cross-validated grid-search over a parameter grid.

  References:

  - GridSearchCV link

Benchmark:

- For the Benchmark Model, we can use Logistic Regression to make the classification and we can measure the f1_score obtained for training and testing to compare with other models that had will be implemented.

- We can choose f1_score as a metric because our dataset is not balance in outcome
  0 ⇒ 1316
  1 ⇒ 684
- Benchmark result:
  Logistic Regression f1_score is around `0.6723404255319149.`

III. Methodology:

Data Preprocessing:

- The dataset is collected from kagg, the shape of the dataset is (2000,9)..
- The dataset is already clean data.
- We will later split the data into training samples(80% of the dataset) and testing samples(20% of the dataset).
- For this dataset there is no need to data preprocessing.

Implementation:

- Our implementation is mainly divided into seven steps:

1. **Exploring the Data:**
    - Exploring the Data using jupyter notebook and verify we need to data preprocessing or not.For this dataset there is no need to data preprocessing.
    - Before split into train and test data check outcome balanced or in balance.whether it is in balanced then we used f1_score.

2. **Split into train and test:**
    - In this we will use `train_test_split` which is available in `sklearn.model selection`.
    - Where we will be dividing the data into train 80% and test splits of 20%.

3. Benchmark model vs other models:

- We can choose our benchmark model as LogisticRegression. Because it is simple and easy to implement.
- We will apply a series of classification algorithms like Logistic Regression, Decision Trees, Random forests etc.
- The next step is to find and compare f1_scores of all the models against the benchmark model.

4. Apply Grid Search Optimization:

- Since we can observe that the model with f1_score is Random Forests with an f1_score of 97.65%.
- We can select Random forest for further optimizing it by applying the same algorithm on a different set of parameters to find the one with the best f1_score.

5. Display results

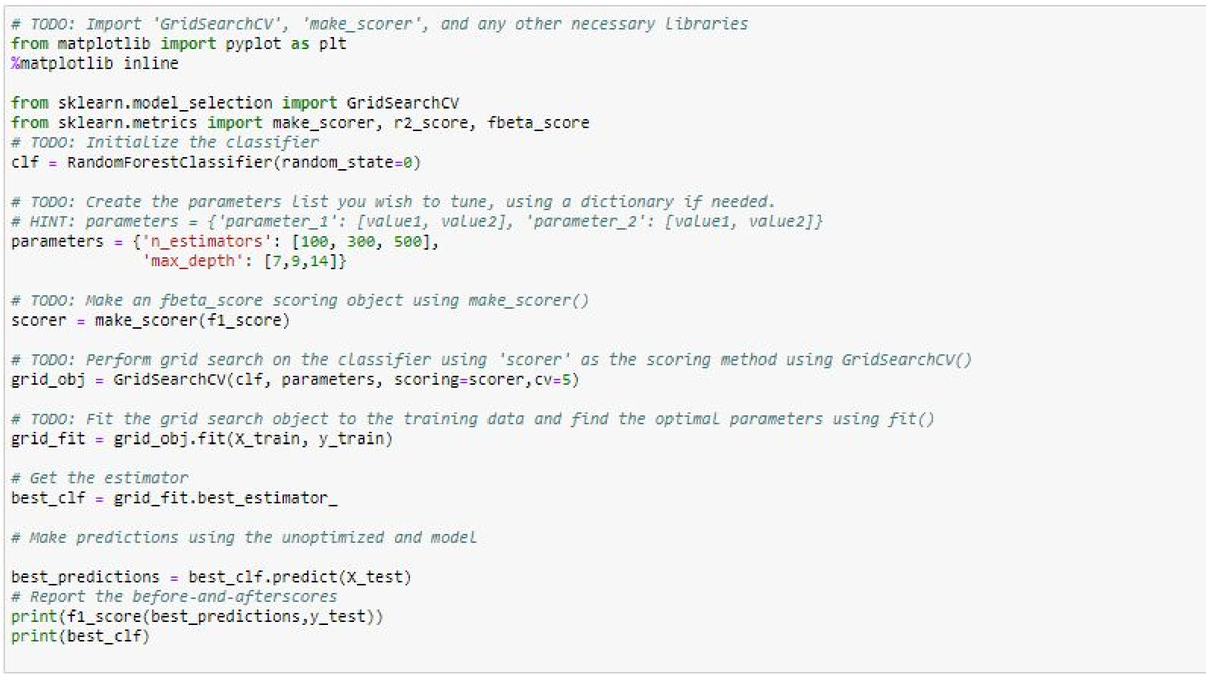
- Here we will apply many supervised learning algorithms and decide which algorithm is best suited for our dataset.
- We can apply the grid search on some chosen algorithms to decide the best parameters among many.

- Finding gridsearchcv optimal f1_score changing parameters to get best optimal f1_score.gridsearchcv (parameters like n_estimator,max_depth..etc) to get best f1_score for this best classification model using gridsearchcv. I faced some more extra time taken to find best f1_score to change parameters to this diabetes machine learning project.

Refinement:

- The algorithm with the highest f1_score is the random forest classifier. To further improve the f1_score the random forest is subject to different parameters by applying grid search on this algorithm
- .

- Whether the parameter are taken in gridsearchcv are n_estimators and max_depth.
-
- When we take n_estimators = [100, 300, 500] and max_depth = [7,9,14] so many combinations are formed and each combination has a score.out of all combinations the max score occur when n_estimators=100,max_depth = 14.



```
# TODO: Import 'GridSearchCV', 'make_scorer', and any other necessary libraries
from matplotlib import pyplot as plt
%matplotlib inline

from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer, r2_score, fbeta_score
# TODO: Initialize the classifier
clf = RandomForestClassifier(random_state=0)

# TODO: Create the parameters list you wish to tune, using a dictionary if needed.
# HINT: parameters = {'parameter_1': [value1, value2], 'parameter_2': [value1, value2]}
parameters = {'n_estimators': [100, 300, 500],
              'max_depth': [7,9,14]}

# TODO: Make an fbeta_score scoring object using make_scorer()
scorer = make_scorer(f1_score)

# TODO: Perform grid search on the classifier using 'scorer' as the scoring method using GridSearchCV()
grid_obj = GridSearchCV(clf, parameters, scoring=scorer,cv=5)

# TODO: Fit the grid search object to the training data and find the optimal parameters using fit()
grid_fit = grid_obj.fit(X_train, y_train)

# Get the estimator
best_clf = grid_fit.best_estimator_

# Make predictions using the unoptimized and model

best_predictions = best_clf.predict(X_test)
# Report the before-and-afterscores
print(f1_score(best_predictions,y_test))
print(best_clf)
```

```
 0.9765625
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
          max_depth=14, max_features='auto', max_leaf_nodes=None,
          min_impurity_decrease=0.0, min_impurity_split=None,
          min_samples_leaf=1, min_samples_split=2,
          min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
          oob_score=False, random_state=0, verbose=0, warm_start=False)
```

Results :

Model Evaluation and validation

- We can observe different f1_scores of all the models, that we applied on the data. In the following table.

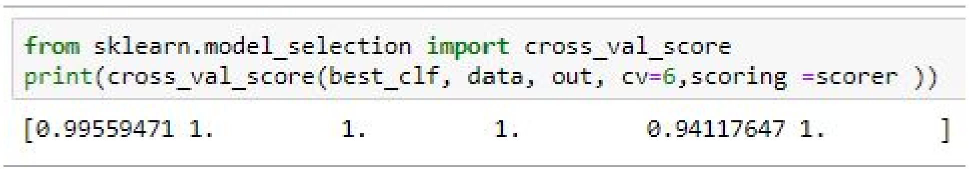
S.No	Supervised Algorithm	f1_Score
1.	Logistic Regression(Benchmark model)	**0.6723404255319149**
2.	SVMClassifier	**0.7723577235772358**
3.	DecisionTreeClassifier	**0.9328063241106719**
4.	RandomForestClassifier	**0.944**

- When we observe the above table. The benchmark model(Logistic Regression) has an f1_score of `0.6723404255319149`

- Among all the above models' random forest has the highest f1_score. Which is `0.944`

- So we applied Grid Search Cross Validation on Random forest and obtained an f1_score of `0.9765625.` Whether the parameter are taken in gridsearchcv are n_estimators and max_depth.When we take n_estimators = [100, 300, 500] and max_depth = [7,9,14] so many combinations are formed and each combination has a score. out of all combinations the max score occur when n_estimators=100,max_depth = 14.

- Which is higher than the unoptimized model.
- Here applying cross valid score on best optimal model(gridsearchcv). The maximum of number of cross_valid_scores of similar by applying cross For this machine learning algorithm to be considered robust because this work on any dataset.where the result are shown below.



```
from sklearn.model_selection import cross_val_score
print(cross_val_score(best_clf, data, out, cv=6,scoring =scorer ))
```

```
[0.99559471 1.         1.         1.         0.94117647 1.        ]
```

Justification:

- The models used in the diabetes based on features were compared and researched, the results showed that all models have good predictive ability, and

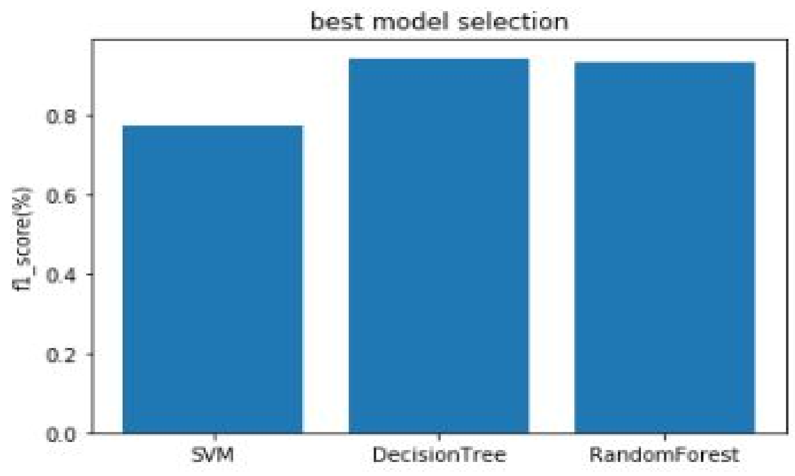
- Logistic Regression has the least f1_score and random forest algorithm model showed the best f1_score on the target variable with the best predictive ability, can be very good in predicting new diabetes.

- The benchmark f1_score is around `0.6723404255319149`
  The optimised model has obtained f1_score `0.9765625`
  Which is better than our benchmark model.

Conclusion:

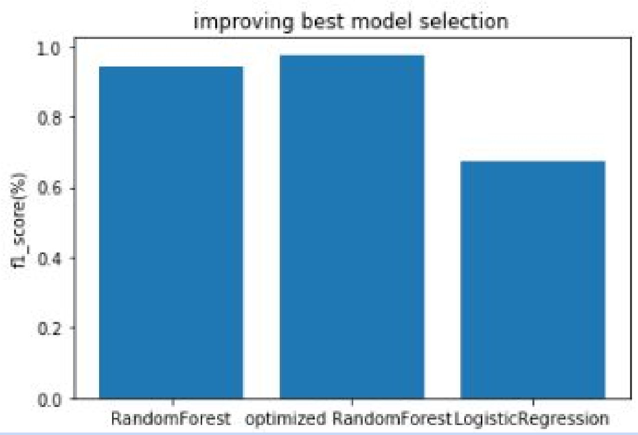
Free-form Visualization

```
labels = ["SVM", "DecisionTree", "RandomForest"]
usage = [SVM_score, RandomForest_score, DecisionTree_score]

y_positions = range(len(labels))
plt.bar(y_positions,usage)
plt.xticks(y_positions, labels)
plt.ylabel("f1_score(%)")
plt.title("best model selection")
plt.show()
```



- In the above visualization, we can see that Random forest Classifier has high f1_score among three model svm,decision tree,randomforest.

```
labels = ["RandomForest","optimized RandomForest ", "LogisticRegression" ]
usages = [max(usage),GridsearchCV_score,LogisticRegression_score]

y_positions = range(len(labels))
plt.bar(y_positions,usages)
plt.xticks(y_positions, labels)
plt.ylabel("f1_score(%)")
plt.title("improving best model selection")
plt.show()
```



- In the above visualization, we can see that improving best model random forest Classifier using gridsearchcv. The gridsearchcv f1_score is slightly greater than random forest.

Reflection:

1. During my process of doing this project, I learnt how to visualize, and understand the data.
2. For this dataset there is no need to data preprocessing.
3. In this we will use `train_test_split` which is available in `sklearn.model selection` to Split into train and test.
4. We can choose our benchmark model as LogisticRegression. Because it is simple and easy to implement.
5. We will apply a series of classification algorithms like Logistic Regression, Decision Trees, Random forests etc.
6. The next step is to find and compare f1_scores of all the models against the benchmark model.
7. We find f1_scores of all the models, than we applied on the data gridsearchcv algorithm to find best classification model f1_score.
8. To find cross_valid_score whether the find best classification model is robust or not.

9. I have come to know how to use the best algorithm in different conditions for the data using appropriate techniques.”sklearn” helped me a lot in knowing a lot about the respective algorithms and their parameters.
10. I am aware of how to find f1_score to classification algorithm.
11. Along with all the above points, I have learnt how to take a data set which is open source in kaggle and applying techniques on it.
12. And also learnt to stick to best techniques to get good results.
13. Finally, I am glad that I can solve a problem and acquire a solution using machine learning concepts.

Improvement:

- In this project, I have evaluated the different Classification algorithm for getting the Insight into diabetes dataset.

- The model is very appropriate to identify whether diabetes or not. In future, research can be used to use a more refined technique to give more f1_score.

- Maybe in future, we can extend the concept of predict diabetes by using a different approach. I.e, collecting data as a training set and labelling it with the corresponding values based on the diabetes.