Introduction

For our study we will use the **Pima Indian dataset**, initially collected by the National "Institute of Diabetes and Digestive and Kidney Diseases". Several constraints were placed on selecting these instances from a more extensive database. In particular, all patients here are females at least 21 years old of Pima Indian heritage. The datasets consist of several medical predictor variables and one target variable, Outcome. Predictor variables include the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

Data Preparation

The dataset contains a large number of missing values which have been saved as the value 0 as shown to the right. Using this will mislead the model. Hence, we will replace the missing values with their medians.

1. Insulin: 374
2. SkinThickness : 227
3. BloodPressure : 35
4. BMI : 11
5. Glucose : 5

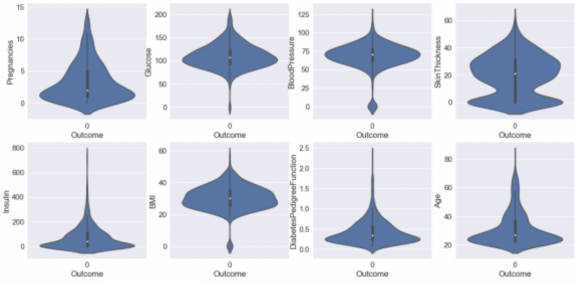
Tts clearly visible from the correlation table below that the Glucose attribute is very important in the outcome prediction as its correlation coefficient is comparatively high at 0.495 followed by BMI, skin thickness, age, pregnancies, blood pressure, diabetes pedigree function and insulin.¶

Skin thickness seems to have a high correlation of 0.57 with BMI and so we drop skin thickness in order to avoid correlation between attributes. Pregnancies also seem to be highly correlated with the age attribute so we drop this as well in order to prevent the effects of multicollinearity on the machine learning algorithm.¶

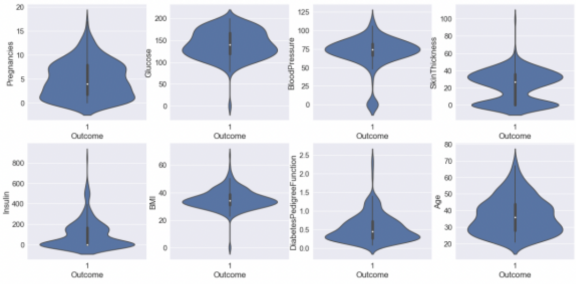
Insulin and blood pressure seems to be very loosely correlated to the outcome so we drop that too. From this, we find that Glucose, BMI, Age and Diabetes Pedigree Function are the most important attributes and hence our Machine learning models will use only these.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.130155	0.209155	0.044593	-0.074531	0.023887	-0.033523	0.544341	0.221898
Glucose	0.130155	1.000000	0.224684	0.215366	0.309058	0.236225	0.138353	0.268910	0.495990
BloodPressure	0.209155	0.224684	1.000000	0.159653	-0.043125	0.285945	-0.001336	0.325306	0.173316
SkinThickness	0.044593	0.215366	0.159653	1.000000	0.204425	0.573227	0.144541	0.063847	0.282844
Insulin	-0.074531	0.309058	-0.043125	0.204425	1.000000	0.157359	0.165705	-0.033698	0.057946
BMI	0.023887	0.236225	0.285945	0.573227	0.157359	1.000000	0.152779	0.027873	0.315606
DiabetesPedigreeFunction	-0.033523	0.138353	-0.001336	0.144541	0.165705	0.152779	1.000000	0.033561	0.173844
Age	0.544341	0.268910	0.325306	0.063847	-0.033698	0.027873	0.033561	1.000000	0.238356
Outcome	0.221898	0.495990	0.173316	0.282844	0.057946	0.315606	0.173844	0.238356	1.000000

To see the distribution of all attributes, we plotted violin plots separately for positive and negative classes. The data for most attributes, except for blood pressure does not seem to follow a normal distribution and hence will require to be centred and scaled.



Violin plot for each parameter for 0 Outcome value



Violin plot for each parameter for 1 Outcome value

Libraries used

The Pandas library was implemented for the purpose of loading the CSV data as a DataFrame and manipulating the data. The NumPy library was used to make calculations and manipulate the data arrays. Matplotlib and Seaborn for plotting. The sklearn library was the most extensively used right from the StandardScaler and pipeline method to using GridSearchCV for hyper-parameter selection and the deployment of all the ML models and implementing their metrics.

Classification Methods

Logistic regression
Parameters selected:
Solver : [‘liblinear’, ‘lbfgs’]
Penalty : [None, ‘l1’, ‘l2’]
max_iterations : [15, 20, 30, 40]

Decision Trees
Parameters selected:
criterion :['gini', ‘entropy']
splitter: ['best', ‘random']
max_depth : [2,4,5,6,7]
min_samples_split : [2,3,4,5,6]
min_samples_leaf: [1,2,3,4,5]
max_features': [None, 'auto', 'sqrt', ‘log2’]

K nearest neighbours
Parameters used:
n_neighbors: [5, 10, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34],
algorithm : ['auto', 'ball_tree', 'kd_tree', 'brute'],
weights : ['uniform', ‘distance']
GridSearchCV was used to find the optimal parameters for all the above models and the results were plotted.

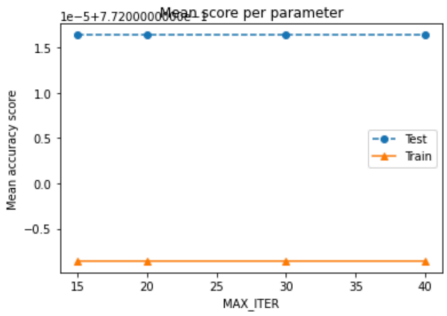
Training and Testing process

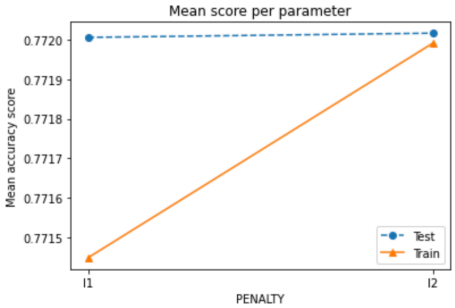
For the purpose of training and testing the cleaned dataset was split using sklearn’s train_test_split method where the training data had about 80% of the shuffled data points while the test data made up 20% of the dataset.

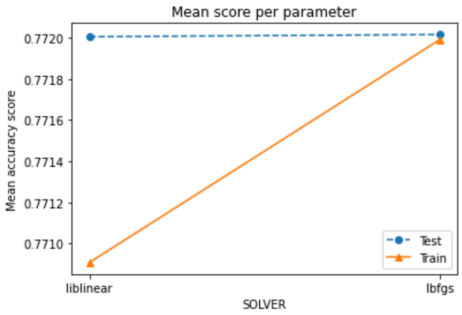
For each algorithm, GridSearchCV was used in order to select the best model parameter using 10 folds and the model with the highest 10 fold average validation accuracy was selected and compared to the training accuracy to reconfirm that the model is not overfitting or underfitting the data. Once confirmed the trained model was used to make predictions on the previously unseen test data and the accuracy, recall, precision, f1-score and confusion matrix is displayed.

Evaluation

Logistic Regression





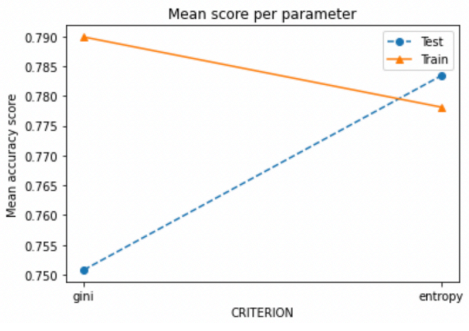


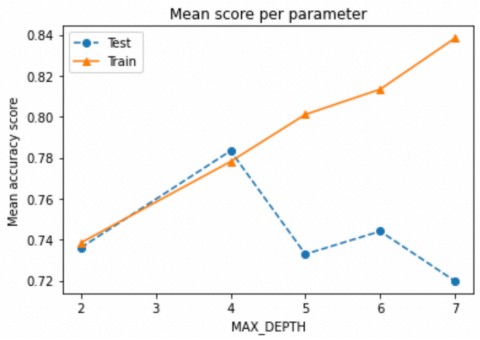
final parameters selected: max_iter = 15, penalty = 'l2', solver = 'lbfgs

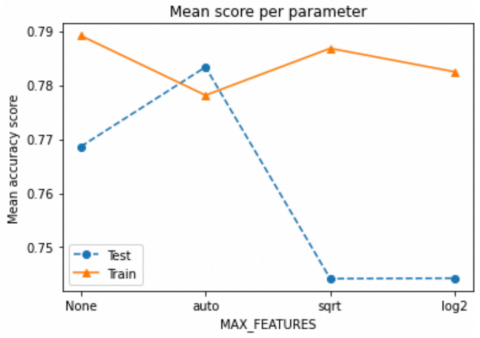
```
Training accuracy:  0.7703583061889251
Average validation accuracy:  0.7752247488101534
[[354  47]
 [ 91 122]]
precision score:  0.7218934911242604
recall score:  0.5727699530516432
F1-score:  0.6387434554973823
```

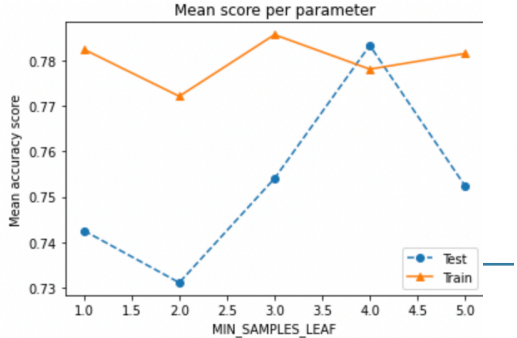
```
Test accuracy:  0.7727272727272727
[[84 15]
 [20 35]]
precision score:  0.7
recall score:  0.6363636363636364
F1-score:  0.6666666666666666
```

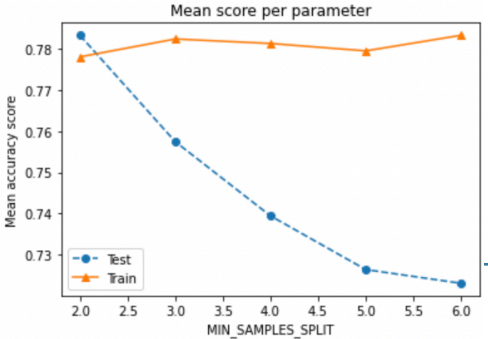
Decision Tree Classifier

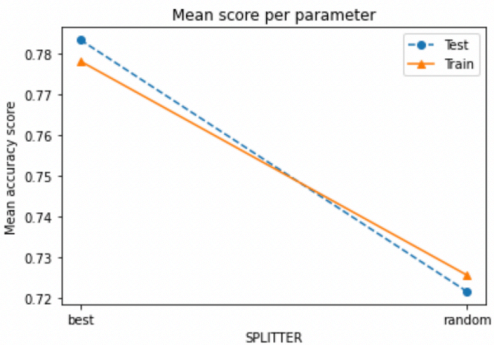










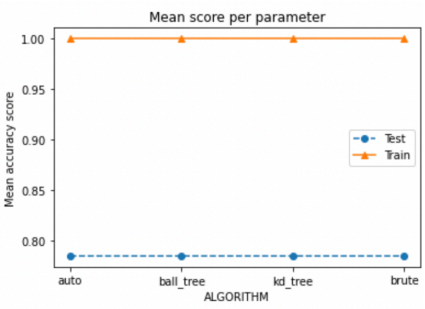


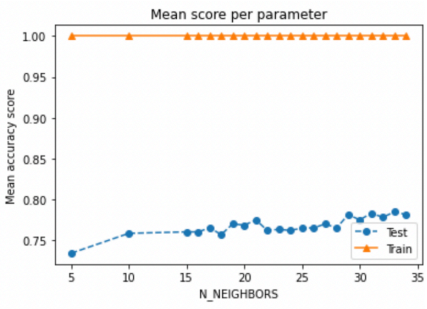
Final parameters selected : criterion = 'entropy', max_depth = 4, max_features = 'auto', min_samples_leaf = 4, min_samples_split = 2, splitter = 'best

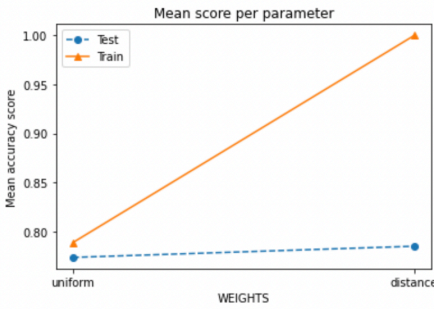
```
Training accuracy:  0.7833876221498371
Average validation accuracy:  0.7491274457958752
[[338  63]
 [ 94 119]]
precision score:  0.6538461538461539
recall score:  0.5586854460093896
F1-score:  0.6025316455696202
```

```
Test accuracy:  0.7597402597402597
[[89 10]
 [27 28]]
precision score:  0.7368421052631579
recall score:  0.509090909090909
F1-score:  0.6021505376344085
```

K Nearest Neighbours Algorithm







Final parameters selected : algorithm = 'auto', n_neighbors = 33, weights ='uniform'

```
Training accuracy:  0.7915309446254072
Average validation accuracy:  0.7735589635113695
[[349  52]
 [ 87 126]]
precision score:  0.7078651685393258
recall score:  0.5915492957746479
F1-score:  0.6445012787723786
```

```
Test accuracy:  0.7597402597402597
[[82 17]
 [20 35]]
precision score:  0.6730769230769231
recall score:  0.6363636363636364
F1-score:  0.6542056074766355
```

Training metrics summary

Model	Training accuracy	10 fold average cross validation accuracy	Precision	Recall	F1 score
Logistic Regression	0.7703	0.7752	0.7218	0.5727	0.6387
Decision tree	0.7833	0.7491	0.6538	0.5586	0.6025
K-nn	0.7915	0.7735	0.7086	0.5915	0.6445

Testing metrics summary

Model	Testing accuracy	Precision	Recall	F1 score
Logistic Regression	0.7727	0.7	0.6363	0.6666
Decision tree	0.7597	0.7368	0.5090	0.6021
K-nn	0.7597	0.6730	0.6363	0.6542

Conclusion

Logistic regression is the more acceptable method among the three because it takes less computational power and has a much higher testing accuracy of nearly 77% and also a higher F1 score compared to the K-nn and Decision tree models which indicates that this model has a similar precision and recall, unlike the others where the difference between the two is a bit higher. Although the training accuracies are higher for the decision trees and K-nn models there is a small amount of disparity between the training and validation accuracies (~2%) indicating slight overfitting which is evident from its lower but not too far off accuracy values on the never before seen test data.

Challenges of the project

The main challenges of the project were mainly selecting the important feature needed for implementing the models which we overcame by using the correlation plot and the second one was while searching for the optimal parameters for the models using grid search. We had to restrict the search to only a few parameters and their corresponding values as we were working on our laptops which are limited in processing power which would result in long processing times.

Task allocation

Aditya Naik: Data cleaning, Logistic Regression model implementation and analysis, Report writing

Reza Fazli: Decision Tree classifier model implementation and analysis, Report writing

Yuzhe Qiu: K-nn model implementation and analysis, Report writing

References

https://www.kaggle.com/uciml/pima-indians-diabetes-database

https://kambria.io/blog/logistic-regression-for-machine-learning/

https://en.wikipedia.org/wiki/Binary_classification

https://www.webmd.com/men/weight-loss-bmi

https://en.wikipedia.org/wiki/Glucose_tolerance_test