BIRMINGHAM CITY University

CMP6202

Artificial Intelligence & Machine Learning

2024–2025

Predicting the presence of diabetes mellitus using supervised learning models

Lewis Higgins - Student ID 22133848

Module Coordinator: Nouh Elmitwally

Contents

Abstract

This report presents a comprehensive study on predicting the presence of diabetes mellitus using supervised learning models. The research integrates two datasets, the Pima Indian Diabetes Database and a Frankfurt diabetes dataset, resulting in a combined dataset of 2,768 samples. Extensive exploratory data analysis revealed key insights into feature relationships and data quality issues, which were addressed through preprocessing techniques including outlier handling, missing data imputation with KNN, and class balancing with SMOTE. Five classification algorithms were developed and evaluated: Random Forest, Support Vector Machine, Logistic Regression, Naïve Bayes, and K-Nearest Neighbours. The models had an initial iteration with reasonably good performance, which was further improved in a second iteration of each model using stratified cross-validation and hyperparameter tuning. K-Nearest Neighbours emerged as the top-performing model, achieving 97.8% accuracy, 96.2% recall, and an F1-score of 96.6 on the unseen test set, with this performance closely followed by Random Forest. The study demonstrates the potential of machine learning in the medical field, while also highlighting the importance of proper data preprocessing and model optimization techniques.

Introduction

Diabetes mellitus, or type 2 diabetes, accounts for 90% of the 4.4 million cases of diabetes in the UK, and it is estimated that there are 1.2 million undiagnosed cases of type 2 diabetes across the country (Diabetes UK, 2024a). The rate of type 2 diabetes per 100,000 individuals is rapidly increasing, with Khan et al. (2020)'s analysis projecting that by 2030, the rate will reach 7,079 per 100,000. Many people with diabetes suffer immensely reduced quality of life, with approximately 50% of patients suffering from peripheral neuropathy (Dhanapalaratnam et al., 2024), an irreversible disability which causes immense pain due to nerve damage from high blood sugar (NHS, 2022), which can occur when the patient was unaware they even had diabetes.

Therefore, it is imperative that systems are put in place to enable the swift diagnosis of diabetes, especially type 2 diabetes given its major prevalence. This can be accomplished by training machine learning models on existing clinical datasets to identify common trends in those with and without type 2 diabetes. This report will document the planning, development and evaluation of multiple machine learning models in their classification of whether individuals have type 2 diabetes based on multiple clinical factors, specifically through the stages of:

- Dataset Identification
- Data Integration
- Data Preprocessing
- Exploratory Data Analysis (EDA)
- Model Development
- Model Evaluation
- Research Conclusions

1.1 Dataset Identification

Machine learning models require large amounts of data to train upon, meaning a dataset must be identified consisting of many rows and features. This project identified two datasets which could be integrated into one larger dataset, the first of which being the well-reputed Pima Indian Diabetes Database (UCI Machine Learning, 2024), downloaded from Kaggle, a platform for students and researchers alike to download and upload datasets and code for research purposes. The data originates from the National Institute of Diabetes and Digestive and Kidney Diseases, who collected this data from Pima Indian[1] women aged 21 and over in hospitals in Phoenix, Arizona, USA, and it has previously seen wide use across academic literature relating to machine learning (AlZu'bi et al. (2023), Zou et al. (2024), Joshi and Dhakal (2021), Hayashi and Yukita (2016)), where other researchers have also aimed to solve the problem of diabetes classification via supervised learning. This dataset contains 768 rows with 9 features.

This project also includes a second dataset, also from Kaggle, that has been previously used in literature by Zou et al. (2024). This dataset (John DaSilva, 2024) is based on data from female patients in Frankfurt, Germany, and includes the same 9 features as the Pima Indian dataset, but includes 2000 rows. By integrating these two datasets into one larger dataset of 2768 rows, it will be possible to give the machine learning models more data to train upon.

Table 1.1 details the 9 features seen in both datasets and their descriptions.

Feature	Description
Pregnancies	The number of pregnancies the patient has had.
Glucose	Plasma glucose concentration over 2 hours in an oral glucose tolerance test.
BloodPressure	Diastolic blood pressure in mm/Hg.
SkinThickness	Triceps skin fold thickness (mm)
Insulin	2-hour serum insulin.
BMI	Body Mass Index, calculated from the patient's weight and height.
DiabetesPedigreeFunction	The product of a function to ascertain the probability of diabetes based on family genetics. (Akmeşe, 2022)
Age	The patient's age.
Outcome	Whether the patient is likely to develop diabetes.

Table 1.1: The features seen in both datasets.

[1] "Pima Indian" refers to a specific Native American ethnic group rather than people from India.

1.2 Supervised learning task identification

As previously mentioned, it is possible for patients to have diabetes without knowing. Therefore, it is paramount that swift and simple diagnosis methods are put in place, which can be achieved through the use of supervised learning classification models. This requires the existence of the "ground truth", which refers to the label given to data that indicates its class (c3.ai, 2024). Within these datasets, the ground truth is present as the 'Outcome' feature, which was used as the target variable for the produced classification models.

Exploratory Data Analysis

This chapter details the EDA processes undertaken with the datasets, including key questions that will be answered by the process, as well as the splitting of the data into training and testing sets.

2.1 Data Integration

The two datasets must first be merged into one to allow for an overall analysis to be performed. This is a simple process because they both contain the same 9 features, and is detailed in Figure 2.1.

```
pima_df = pd.read_csv("Data/pima.csv")
pima_df.shape
```
✓ 0.0s
```
(768, 9)
```

```
frankfurt_df = pd.read_csv("Data/frankfurt.csv")
frankfurt_df.shape
```
✓ 0.0s
```
(2000, 9)
```

```
df = pd.concat([pima_df, frankfurt_df], axis = 0, ignore_index = True)
df.shape
```
✓ 0.0s
```
(2768, 9)
```

Figure 2.1: Integrating the two separate datasets into one larger dataset.

2.2 Question identification and assumptions

The key factors involved in the diagnosis of diabetes are critical to understand, which can be solved through EDA on these datasets. It is possible to make various assumptions based on topical background research of each of the features in the dataset, detailed in Table 2.1

Feature	Research-based assumptions
Pregnancies	Approximately 13.4% of pregnant women develop a temporary condition known as Gestational Diabetes Mellitus (GDM), which typically subsides after birth (Adam et al., 2023). However, research by (Dennison et al., 2021) indicates that 33% of women who develop GDM will go on to develop permanent diabetes mellitus within 15 years. Therefore, it is assumed that pregnancies will positively correlate with the diabetes outcome. It is also expected that pregnancies should naturally positively correlate with age.
Glucose	Glucose concentrations are an enormous factor in the diagnosis of diabetes mellitus, being one of the main metrics used to certify the condition, where results over 200mg/dL mean an absolute diagnosis[1] (Aftab et al., 2021). It is therefore assumed that the glucose concentrations will be one of the strongest influences of the outcome, and that it will also correlate heavily with insulin levels.
BloodPressure	Diastolic blood pressure (DBP) does influence the diagnosis of diabetes mellitus, as 56.2% of recently diagnosed patients presented with elevated DBP in Nelaj et al. (2023)'s limited study of 126 patients, but it is not a decisive factor by itself. Therefore, it is assumed that there will be some correlation between DBP and the outcome, but not as major as other factors like plasma glucose levels.
SkinThickness	It is a frequent assumption even non-academically that people who weigh more, and by consequence have higher skin thickness in certain areas such as the triceps, have a higher risk of developing conditions like type 2 diabetes. This is backed by a study by Ruiz-Alejos et al. (2020), which found strong associations between skin thickness and diabetes mellitus, as well as high blood pressure. Therefore, it is assumed that there will be a strong correlation between tricep skin thickness and the outcome, as well as an expectation of strong correlations between thickness, BMI and blood pressure.
Insulin	Diabetes mellitus is directly associated with insulin deficiency, and as such, it is assumed that this factor will be the strongest influence in the outcome. This is because 2-hour serum insulin tests, as used in this dataset, are frequently part of HOMA-IR[2] assessments.

[1]The other main metric is insulin deficiency, meaning that the patient could have glucose levels lower than 200mg/dL and still be diagnosed if they are instead insulin deficient. (Aftab et al., 2021).

[2]Homeostasis Model Assessment of Insulin Resistance, used to measure insulin resistance (Tahapary et al., 2022), which can be used in both type 1 and type 2 diabetes diagnosis (Khalili et al., 2023).

BMI	BMI is likely to be a significant factor in the outcome, which is backed by previous academic studies indicating that 71% of studied individuals showed increases in BMI prior to diagnosis (Donnelly et al., 2024). Additionally, BMI is used in insulin resistance measurement assessments, which are key assessments in diabetes diagnosis, meaning that it is a safe assumption that BMI will be a large factor in the outcome.
DiabetesPedigree-Function	People are more likely to develop diabetes mellitus if there is a family genetic history of the condition, though it is not directly caused by any one particular gene (Diabetes UK, 2024b). With the pedigree function aiming to quantify the inheritance probability, it can be assumed that it will likely correlate heavily with the outcome.
Age	Suastika et al. (2012) studied the effects of age as a risk factor for diabetes mellitus, finding that many natural associated factors of ageing including increases in body fat and decreases in lipid metabolism had considerable influence on the development of insulin resistance and diabetes mellitus by consequence. Therefore, it is likely that there will be a noticeable correlation between a patient's age and the outcome.

Table 2.1: Research-based assumptions prior to any EDA.

Based on these assumptions, the questions that this EDA process aims to answer are:

ID	Research-based assumptions
1	Are there any missing values or values that are not physically possible?
2	Are there any significant outliers?
3	Is the dataset evenly balanced in terms of the outcome? If not, what should be done?
4	Does the rate of diabetes positively correlate with the amount of pregnancies a woman has had?
5	Does the amount of pregnancies influence any of the other features?
6	What is the distribution of blood glucose levels in patients with and without diabetes?
7	Does BMI influence glucose levels?
8	Is diastolic blood pressure a worthwhile diagnosis method in this dataset?
9	Is the average skin thickness of those with diabetes actually higher than those without?
10	How does the relationship between insulin and glucose change between those with and without diabetes?

Table 2.2: The questions that this EDA process aims to answer.

2.3 Splitting the dataset

It is good practice to first split the data into training and testing sets before performing exploratory data analysis to avoid conceptual overfitting, also known as data leakage. Conceptual overfitting occurs when insights gained from the entire dataset influence model development decisions, which may eventually lead to actual overfitting. By excluding the training data from the analysis, it effectively simulates a real-world environment where the data being given to the model is not known, even to its developers.

Splitting the data is a mandatory process when developing supervised learning models, primarily for the prevention of overfitting. Overfitting is a significant challenge in machine learning, where models can perform exceptionally well on their original training data but are unable to generalize to unseen data, making them unsuitable for deployed use. The training set must consist of a large portion of the data so that the model has enough information to analyse and discover trends within, whereas the testing set is a smaller, unseen remainder of the data that the model's predictions can be evaluated against using various metrics. A key point of determination is the proportions of the dataset that should go in each set - there is no 'one-size-fits-all' percentage that can provide the best results for every possible dataset (Sivakumar et al., 2024), and factors such as the size of the dataset play a large part in this. Most commonly, splits are either 70:30 or 80:20 for training and testing sets respectively.

The integrated dataset for this project is 2,768 rows. This is considered to be a small dataset, and as such, it will be best to maximize the size of the training split, so a split of 80% training and 20% testing was used, visualized in Figure 2.2.

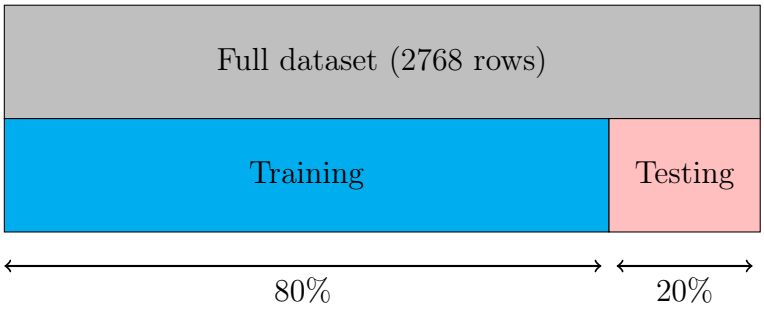


Figure 2.2: A visual representation of the train/test split.

To accomplish this, the data must first be split into X and y tables, where X consists of the eight features, and y is the target variable. After the data is split to X and y, it can be split into training and testing sets through Scikit-Learn's "train_test_split" method, as depicted in Figure 2.3

```
X = df.drop(columns = "Outcome", axis = 1)
y = df["Outcome"]
```
✓ 0.0s

```
print(X.shape)
print(y.shape) # No columns because y is now a Series consisting only of the Outcome column.
```
✓ 0.0s

```
(2768, 8)
(2768,)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```
✓ 0.0s

```
print(f"X_train: {X_train.shape}")
print(f"y_train: {y_train.shape}")
print(f"X_test: {X_test.shape}")
print(f"y_test: {y_test.shape}")
```
✓ 0.0s

```
X_train: (2214, 8)
y_train: (2214,)
X_test: (554, 8)
y_test: (554,)
```

Figure 2.3: Splitting the data at an 80:20 ratio.

By default, this method will first shuffle all rows in the dataset before splitting it, which introduces an element of randomness which can damage reproducibility. To combat this, the "random_state" parameter can be set to ensure that the same shuffle will occur every time.

When performing EDA, the Outcome column will be necessary to the analysis, so a deep copy[3] of X_train was made with y_train (the Outcome column) being added to it, merging the two back into a full training set, shown in Figure 2.4.

```
# It's important to make a deep copy, so that none of the data is tampered with.
fullTrainSet = X_train.copy(deep = True)
fullTrainSet["Outcome"] = y_train.values
```
✓ 0.0s

Figure 2.4: Duplicating X_train and adding the Outcome column for EDA.

[3]A complete copy rather than a pointer to the original data

2.4 EDA process and results

In the interest of report conciseness, the full EDA process and results accompany this report as Appendix A. They were performed using the Seaborn Python library, which uses Matplotlib to produce convenient visualisations to answer each question posed in Table 2.2. The topics and issues identified during the process are solved in Section 3.3.

2.5 EDA conclusions

The extensive EDA conducted in Appendix A provided clear answers to each question posed in Table 2.2, which are detailed below in Table 2.3.

ID	EDA-based answer
1	There are missing values in the dataset that were not originally presented as such; they were instead 0s in columns where this would be physically impossible. After converting the impossible 0s to missing values with NumPy, it was established that there were 18 missing Glucose values, 125 missing BloodPressure values, 800 missing SkinThickness values, 1330 missing Insulin values, and 39 missing BMI values.
2	The analysis identified significant outliers across various features using box plots. Particularly extremely outliers were noted in the SkinThickness, BMI, and Insulin columns. These outliers could affect data scaling and model performance, and will need to be transformed in data cleaning.
3	The dataset is imbalanced, with 65.5% of rows in the training set having an outcome of 0, while the remaining 35.5% had an outcome of 1.
4	The EDA did show a somewhat positive correlation between the number of pregnancies and diabetes outcomes, although pregnancies alone are not decisive in predicting diabetes.
5	Pregnancies are found to correlate with age, as expected, but do not significantly influence other features.
6	There is a clear and significant distinction in glucose levels between diabetic and non-diabetic individuals, with higher levels being mostly present in those with diabetes, making it a strong indicator.
7	A slight positive correlation is observed between BMI and glucose levels, indicating that higher BMI could be associated with higher glucose levels.
8	Blood pressure shows some correlation with diabetes outcomes but is not a strong standalone diagnostic factor. However, like pregnancies, previously mentioned academic research specifies that it is still a useful feature to keep.
9	Individuals with diabetes tend to have higher skin thickness on average, although this feature contains significant high outliers in both individuals with and without diabetes.
10	The relationship between insulin and glucose varies between diabetic and non-diabetic individuals, with diabetic individuals showing higher glucose levels as a result of lower insulin levels, which is expected of type 2 diabetes.

Table 2.3: Answers to the questions after the completion of the EDA process.

Overall, the EDA process conducted in Appendix A has been highly beneficial for many reasons. Firstly, it identified critical data issues, such as the abundance of missing values and outliers, which are essential to address to improve model accuracy and reliability. By recognizing these issues early, the EDA ensures that data preprocessing steps can be effectively planned to mitigate their impact. Secondly, the EDA highlighted significant correlations between features and the diabetes outcome, such as the strong influence of glucose levels and BMI on diabetes diagnosis. Understanding these relationships helps in feature selection and engineering, which are crucial for building effective predictive models. Additionally, the EDA revealed class imbalances in the dataset, guiding the development of strategies to handle this imbalance during model training to prevent biased predictions.

The insights gained from the EDA therefore provide a solid foundation for informed decision-making in the experimental design and model development, ensuring that the models are trained on clean, balanced, and relevant data, ultimately enhancing their predictive performance. This is especially crucial given that the models would be used in the medical field, where incorrect predictions could have life-changing ramifications.

Experimental Design

This chapter details the planned algorithms to be leveraged against this dataset, as well as the metrics to evaluate them. Furthermore, leveraged data preprocessing techniques are deeply explored, as well as some potential limitations relating to their use.

3.1 Identification of chosen algorithms

Five classification algorithms will be leveraged on this dataset, these being Random Forests, Support Vector Machines, Logistic Regression, Naïve Bayes, and K-Nearest Neighbours. These were selected due to their previous use in literature based on these datasets, and also due to their high prevalence across many fields. Detailed descriptions of each algorithm, including their mathematical formulae, can be found in Appendix B.

3.2 Identification of appropriate evaluation techniques

When evaluating classification models, it is crucial to select appropriate metrics that provide a comprehensive understanding of the model's performance. For this project, three key evaluation metrics will be used: accuracy, recall, and F1 score. By using these three metrics in combination, we can gain a comprehensive understanding of our classification models' performance. Accuracy provides an overall measure of correctness, recall ensures positive cases are not missed, and the F1 score offers a balanced view that accounts for both precision and recall. This multi-metric approach will allow for a detailed evaluation of the five models. Detailed descriptions of each metric and how it is calculated can be found in Appendix C.

3.3 Data Cleaning and Pre-processing Transformations

3.3.1 Data encoding

Data encoding was not necessary in this dataset as all features were already numeric. However, a detailed description of typical encoding processes can be found in Appendix D.

3.3.2 Data cleaning

The EDA revealed the necessity for data cleaning in this dataset through the transformation of outliers and imputation of missing values.

Outlier handling

It was observed that all columns had outliers, with some having particularly extreme outliers that caused massive variance in the data and the visualisations produced from it. To address these, data was constrained to be within the 1st and 99th percentiles, and any data outside of these would be increased or decreased to one of these boundaries.

```python
# Transforming rows under the 1st percentile or over the 99th percentile.
for column in X_train.columns:
    lowerBound = X_train[column].quantile(0.01)
    upperBound = X_train[column].quantile(0.99)
    # Where data in the column is less than the 1st percentile or more than the 99th,
    # transform it to the percentile.
    X_train[column] = np.clip(X_train[column], lowerBound, upperBound)
    # np.clip will make values lower than the lower bound into the lower bound,
    # and values over the upper bound into the upper bound.
```

✓ 0.0s

Figure 3.1: Using np.clip to transform values less than or greater than the 1st and 99th percentiles.

Missing data imputation

After handling the outliers in the data, missing data can be imputed. The selected method for this was KNN imputation, which uses the KNN algorithm to aggregate data from the k nearest data points and imputes the average of these points in place of the missing value (TrainInData, 2024). This method can be very useful as it will not impute many rows of the same number as mean or median imputation would do, and instead imputes reasonable values based on similar rows, which is especially beneficial given the high correlations between some of the missing data (such as Insulin) and data that is mostly present (such as Glucose).

```python
# Creates a KNN Imputer that aggregates data from the 6 nearest neighbours to fill missing
# data across the previously identified columns.
imputer = KNNImputer(n_neighbors = 6)

# Fit the imputer on the training data and transform missing values.
X_train = imputer.fit_transform(X_train)
# This converts X_train to a NumPy array, thereby removing the column names.
# This is solved by converting it back to a DataFrame with the column names
# of the original X.
X_train = pd.DataFrame(X_train, columns = X.columns)

# The imputer is not fitted to the testing set, and instead only transforms
# missing rows within it.
X_test = imputer.transform(X_test)
# As with X_train, the column names are passed back by converting the
# NumPy array to a DataFrame with the column names of the original X.
X_test = pd.DataFrame(X_test, columns = X.columns)

print(X_train.isna().sum())
print(X_test.isna().sum())
```

✓ 0.0s

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

Figure 3.2: Using KNN imputation to resolve missing data.

K was set to 6 in the imputer to increase the amount of data each imputed point is based on, but also to not average too many data points, as the data still varies despite containing many less outliers.

3.3.3 Data scaling

After imputing missing data, scaling can occur. Algorithms such as KNN and SVMs rely heavily on the distance between data points, meaning they can be heavily skewed by large distances between real numbers. Due to the high variance in the dataset as many features use different scales (Age, Insulin, SkinThickness, and BMI for example), it was decided that standardisation will be the scaling procedure. Standardisation uses the formula in Equation 3.1 to scale data so that it has a mean of 0 and a standard deviation of 1.

$$Z = \frac{X - \mu}{\sigma} \tag{3.1}$$

Z is the standardized value (also known as "z-score")
X is the original value of the feature
μ is the mean of the feature
σ is the standard deviation of the feature

```
# Instantiate the scaler.
sc = StandardScaler()

# Fit the scaler to X_train and transform data in it.
X_train = sc.fit_transform(X_train)
# As with the imputer, the data can be made back into a
# DataFrame with column names.
X_train = pd.DataFrame(X_train, columns = X.columns)

# Only transform the testing data based on the fit from
# the training set.
X_test = sc.transform(X_test)
# As with the imputer, the data can be made back into a
# DataFrame with column names.
X_test = pd.DataFrame(X_test, columns = X.columns)


X_train.head()
```

✓ 0.0s Open 'X_train' in Data Wrangler

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	0.069906	-0.422956	-0.706207	1.624799	-0.532573	0.344790	1.379999	-0.619707
1	-0.842302	-0.127229	-1.212881	0.759820	-0.585500	0.124244	-0.667264	-0.876417
2	-0.538233	0.037064	-2.057339	0.327330	0.166058	1.418111	0.156092	-0.619707
3	-0.234164	0.628518	-0.115086	0.615657	-0.001543	-0.360956	0.922226	-0.534137
4	1.286183	-1.244421	0.138251	0.219208	-0.950693	0.859396	-0.044184	0.492705

Figure 3.3: Using a StandardScaler to standardise the data.

3.3.4 Data balancing

It was also established during EDA that the dataset is imbalanced, mostly containing rows without diabetes. Therefore, the Synthetic Minority Oversampling Technique (SMOTE) was used to generate synthetic data to balance the classes. SMOTE also uses the KNN algorithm to generate data, but randomly selects nearest neighbours and generates data in-between the selected sample and the neighbour (TrainInData, 2023). This is repeated until the classes are balanced.

```
# Instantiate SMOTE, to be applied on the training set.
smote = SMOTE()

# View the original class imbalance.
print(y_train.value_counts())

# Fit SMOTE to the training set, and also generate data to balance it.
X_train, y_train = smote.fit_resample(X_train, y_train)

# We can now see that the classes are balanced.
y_train.value_counts()
```

✓ 0.0s

```
Outcome
0    1449
1     765
Name: count, dtype: int64

Outcome
1    1449
0    1449
Name: count, dtype: int64
```

Pre-SMOTE imbalance

Post-SMOTE

Figure 3.4: Using SMOTE to balance the training set.

SMOTE is not used to balance the testing set, as the testing set should be exclusively real data so that an accurate representation of the model's ability to predict real data can be gathered (Ozbun, 2021).

3.4 Limitations and Options

A key limitation in this design comes from the datasets themselves; they contain colossal amounts of missing data, with almost half of the Insulin values missing. With Insulin being such an integral part of type 2 diabetes diagnosis, the issue of having so much missing data cannot be overlooked. While it was addressed using KNN imputation, the synthetic data generated by this approach could possibly be of poor quality.

Furthermore, SMOTE was used to balance the classes. While SMOTE is a useful tool, it can introduce noise to a dataset. Additionally, SMOTE was performed after the KNN imputation, meaning that a substantial amount of the dataset was then synthetic data.

In future projects, it would be best to use more organised and clean datasets, where such substantial preprocessing measures would not be necessary. Furthermore, the features in the dataset are all numerical. While this is good for the development of machine learning models, it did not allow for the demonstration of encoding techniques, which were instead described in Appendix D.

Model Development

This chapter details the training and evaluation processes of the original produced models before any iterative improvements such as hyperparameter tuning.

4.1 Predictive modelling process

After the extensive EDA and preprocessing of the data, including the transformation of outliers and the imputation of missing data, the refined training set consisting of 80% of the data could then be used to train the five models. No features were removed, as all were deemed to be relevant in diabetes diagnosis from topical research and the exploration of the dataset itself.

While the inner workings of each algorithm differ immensely, Scikit-Learn allows the fitting and prediction of all models to be uniform, using "fit()" methods to fit each model to the training set, and "predict()" methods to run predictions based on the fit. Figure 4.1 shows the code used to fit the models.

```
# Creates 100 decision trees and aggregates their classifications, using the one
# found most commonly across them.
rf = RandomForestClassifier(n_estimators = 100, random_state = 42)

# Finds the optimal hyperplane to seperate classes by.
svc = SVC(random_state = 42)

# Computes a weighted sum of all features and maps it to between 0 and 1 using a
# sigmoid function. Rows over 0.5 are classified as 1, and rows under are classified as 0.
lr = LogisticRegression(random_state = 42)

# RF, SVC and LR have random elements to them. To ensure reproducibility, the random_state is
# set to 42.

# Calculates the probability of diabetes based on the combination of features
# and classifies based on the probability.
nb = GaussianNB()

# Classifies rows based on the 3 nearest data points.
knn = KNeighborsClassifier(n_neighbors = 3)

# Creating a list of the models so that they can be fitted in a for-loop.
models = [rf, svc, lr, nb, knn]

# Fits each model to the training dataset, where it will learn correlations and trends.
for model in models:
    model.fit(X_train, y_train)
✓ 0.3s
```

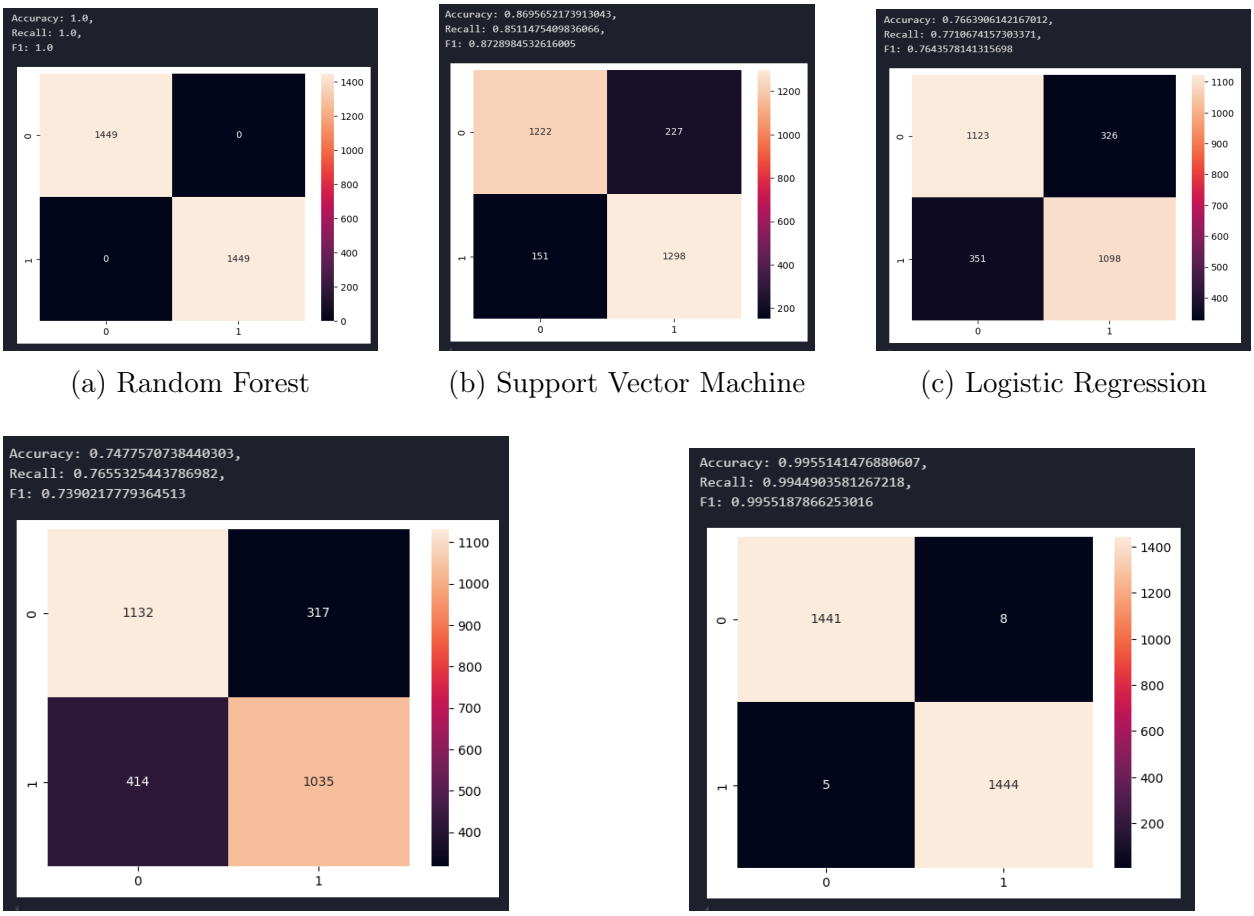
Figure 4.1: The code to fit each of the five models.

4.2 Results on seen data

To evaluate each model, the metrics previously discussed in Section 3.2 were logged, but confusion matrices were additionally produced, which show the amounts of data each model correctly and incorrectly predicted, divided into true positives, false positives, false negatives and true negatives.

```
# Predict on the training set with all five models.
rf_seen_pred = rf.predict(X_train)
svc_seen_pred = svc.predict(X_train)
lr_seen_pred = lr.predict(X_train)
nb_seen_pred = nb.predict(X_train)
knn_seen_pred = knn.predict(X_train)

# Save a list of the predictions to iterate through.
predictions = [rf_seen_pred, svc_seen_pred, lr_seen_pred,
               nb_seen_pred, knn_seen_pred]

# Iterate through each model, outputting it's accuracy,
# recall and F1-Score, also producing a confusion matrix.
for pred in predictions:
    acc = accuracy_score(pred, y_train)
    recall = recall_score(pred, y_train)
    f1 = f1_score(pred, y_train)
    cm = confusion_matrix(y_train, pred) # Confusion matrix takes parameters in the inverse order.

    # Output the metrics.
    print(f"Accuracy: {acc},\nRecall: {recall},\nF1: {f1}")

    # Create the matrix.
    sns.heatmap(cm, annot = True, fmt = "g")
    # Each matrix needs to be individually printed. If this line
    # wasn't used, Seaborn would try to produce one overall matrix.
    plt.show()

✓ 0.7s
```

Figure 4.2: The code to predict with, and output the metrics and confusion matrices of each model on the training set.



(a) Random Forest (b) Support Vector Machine (c) Logistic Regression

(d) Naïve Bayes (e) KNN

Figure 4.3: The confusion matrices and evaluation metrics of each model on seen data.

Model	Accuracy	Recall	F1 Score
Random Forest	100%	100%	100
SVM	86.9%	85.1%	87.2
LR	76.6%	77.1%	76.4
NB	74.7%	76.5%	73.9
KNN	99.5%	99.4%	99.5

Table 4.1: The metrics of each algorithm on the seen training data.

4.2.1 Overall findings

Random Forest and KNN showed the best performance on the seen data, with perfect or nearly perfect scores across all metrics. SVM demonstrated good performance, while Logistic Regression and Naïve Bayes were the least effective. It's important to note that performance on seen data doesn't necessarily translate to good performance on unseen data, and further evaluation using the unseen testing set is crucial to assess the models' true predictive capabilities.

Evaluation and further improvements

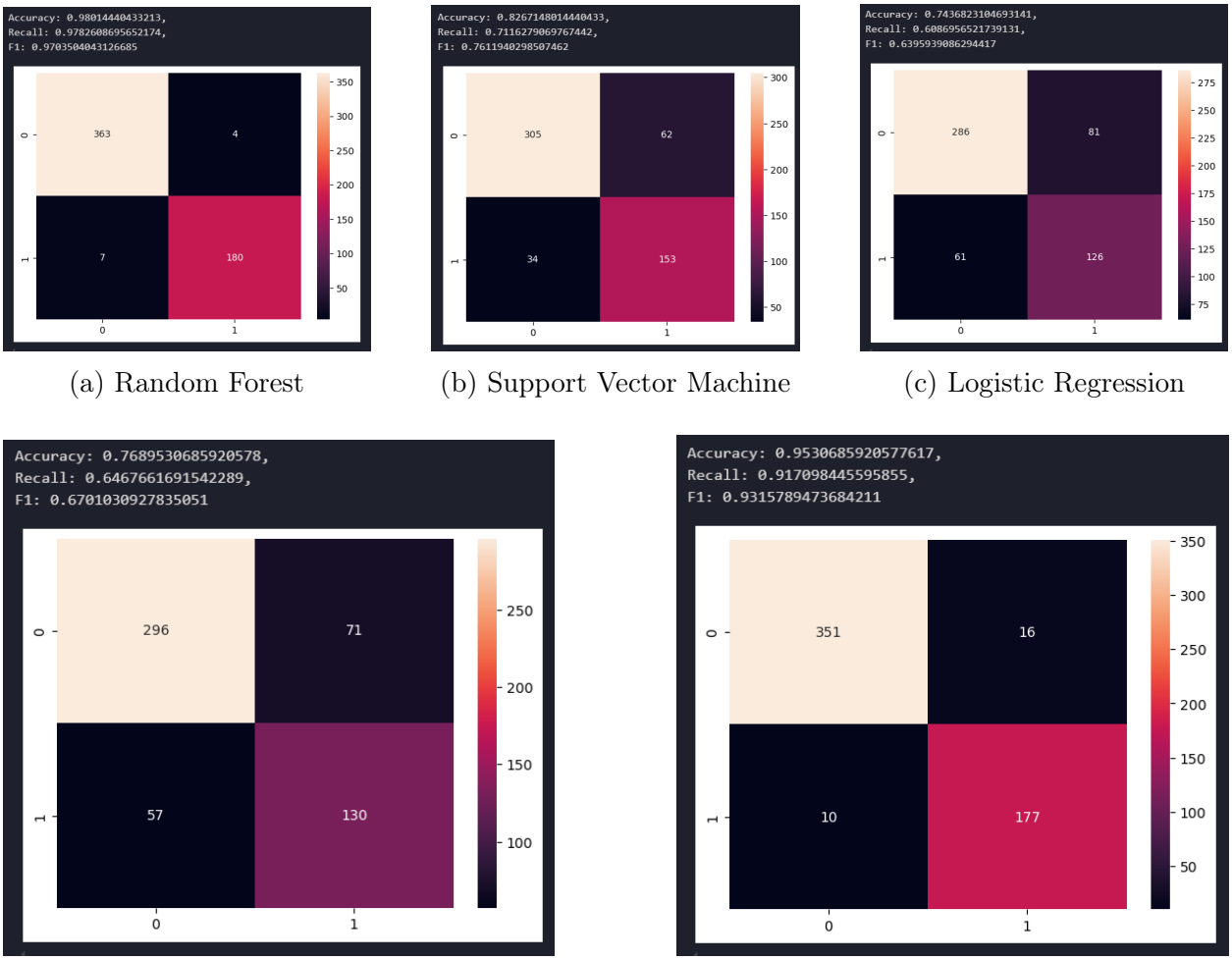
This chapter details the extensive evaluation of each model, as well as iterative improvements that were made to enhance their performance.

5.1 Initial results on unseen data

The models were evaluated against the testing set which they have not previously seen, so that an analysis of how they would be expected to perform on real data could be gathered.

```
# Predict on the testing set with all five models.
rf_unseen_pred = rf.predict(X_test)
svc_unseen_pred = svc.predict(X_test)
lr_unseen_pred = lr.predict(X_test)
nb_unseen_pred = nb.predict(X_test)
knn_unseen_pred = knn.predict(X_test)

# Save a list of the predictions to iterate through.
predictions = [rf_unseen_pred, svc_unseen_pred, lr_unseen_pred,
               nb_unseen_pred, knn_unseen_pred]

# Iterate through each model, outputting it's accuracy,
# recall and F1-Score, also producing a confusion matrix.
for pred in predictions:
    acc = accuracy_score(pred, y_test)
    recall = recall_score(pred, y_test)
    f1 = f1_score(pred, y_test)
    cm = confusion_matrix(y_test, pred) # Confusion matrix takes parameters in the inverse order.

    # Output the metrics.
    print(f"Accuracy: {acc},\nRecall: {recall},\nF1: {f1}")

    # Create the matrix.
    sns.heatmap(cm, annot = True, fmt = "g")
    # Each matrix needs to be individually printed. If this line
    # wasn't used, Seaborn would try to produce one overall matrix.
    plt.show()
```

Figure 5.1: The code to predict with, and output the metrics and confusion matrices of each model on the testing set.



(a) Random Forest (b) Support Vector Machine (c) Logistic Regression

(d) Naïve Bayes (e) KNN

Figure 5.2: The confusion matrices and evaluation metrics of each model on unseen data.

Model	Accuracy	Recall	F1 Score
Random Forest	98%	97.8%	97
SVM	82.6%	71.1%	76.1
LR	74.3%	60.8%	63.9
NB	76.8%	64.6%	67
KNN	95.3%	91.7%	93.1

Table 5.1: The metrics of each algorithm on the unseen testing data.

5.1.1 Overall findings

In conclusion, Random Forest and KNN emerged as the top-performing models, with Random Forest slightly edging out KNN. SVM showed moderate performance, while Logistic Regression and Naïve Bayes were the least effective for this particular diabetes classification task.

5.2 Final model results

Machine learning is an iterative process, where it is unlikely that the first model will achieve excellent results. Therefore, an additional iteration of the models was produced using stratified cross-validation and detailed hyperparameter tuning. Detailed descriptions of how this was accomplished can be found in Appendix D, with the results of these improved iterations shown in Figures 5.3 through 5.7.

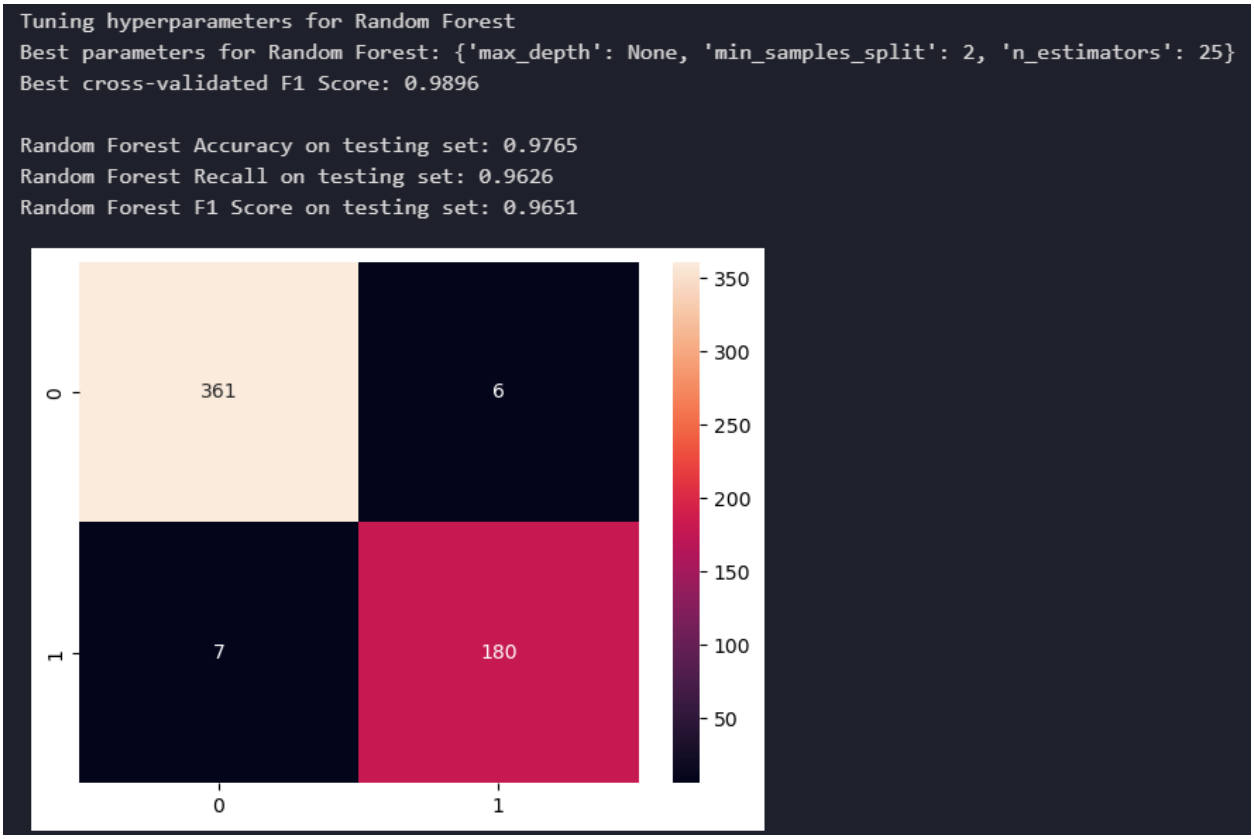


Figure 5.3: The optimal parameters, cross-validated F1 Score, evaluation metrics, and confusion matrix of the Random Forest model.

```
Tuning hyperparameters for Support Vector Classifier
Best parameters for Support Vector Classifier: {'C': 10, 'gamma': 'auto', 'kernel': 'rbf'}
Best cross-validated F1 Score: 0.9037

Support Vector Classifier Accuracy on testing set: 0.8574
Support Vector Classifier Recall on testing set: 0.8503
Support Vector Classifier F1 Score on testing set: 0.8010
```

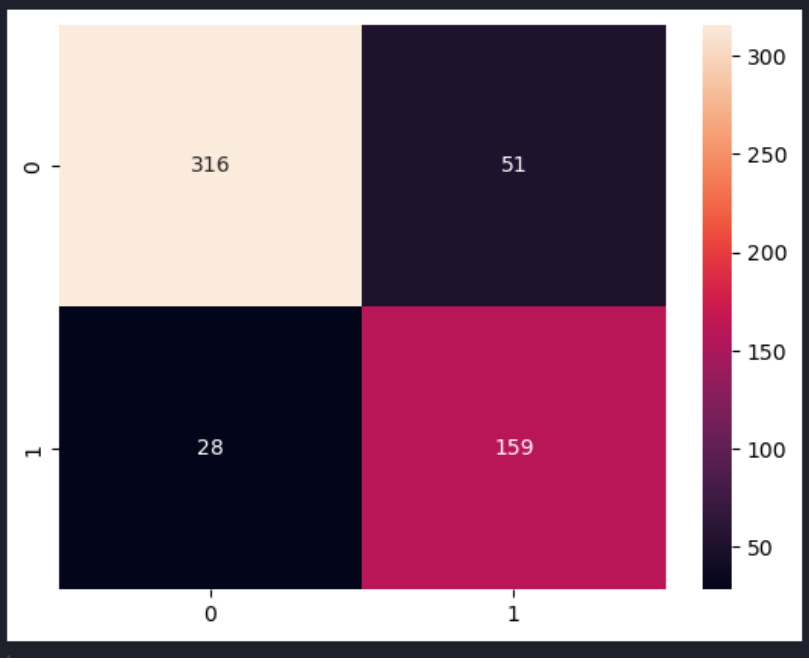


Figure 5.4: The optimal parameters, cross-validated F1 Score, evaluation metrics, and confusion matrix of the Support Vector Machine model.

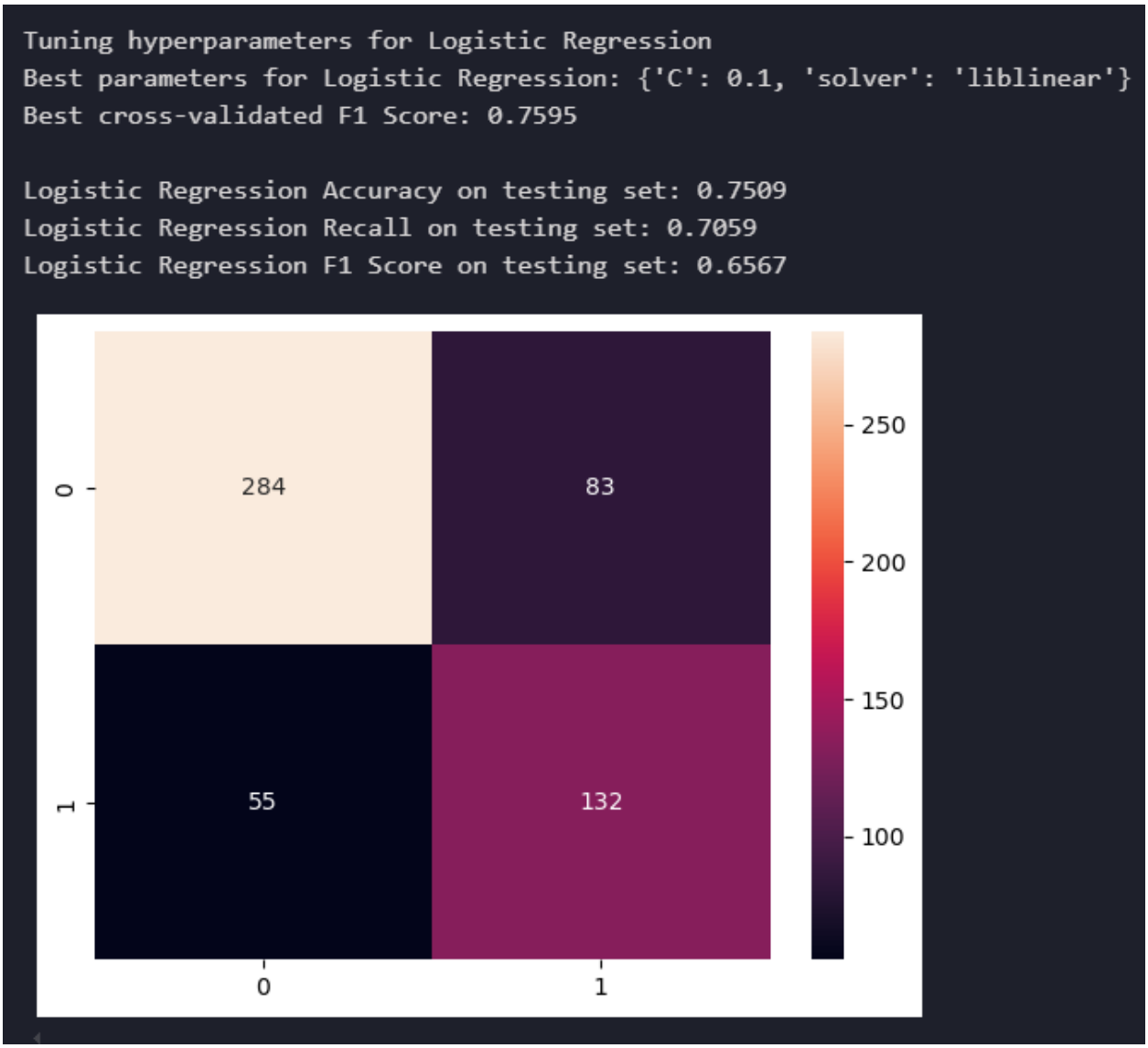


Figure 5.5: The optimal parameters, cross-validated F1 Score, evaluation metrics, and confusion matrix of the Logistic Regression model.

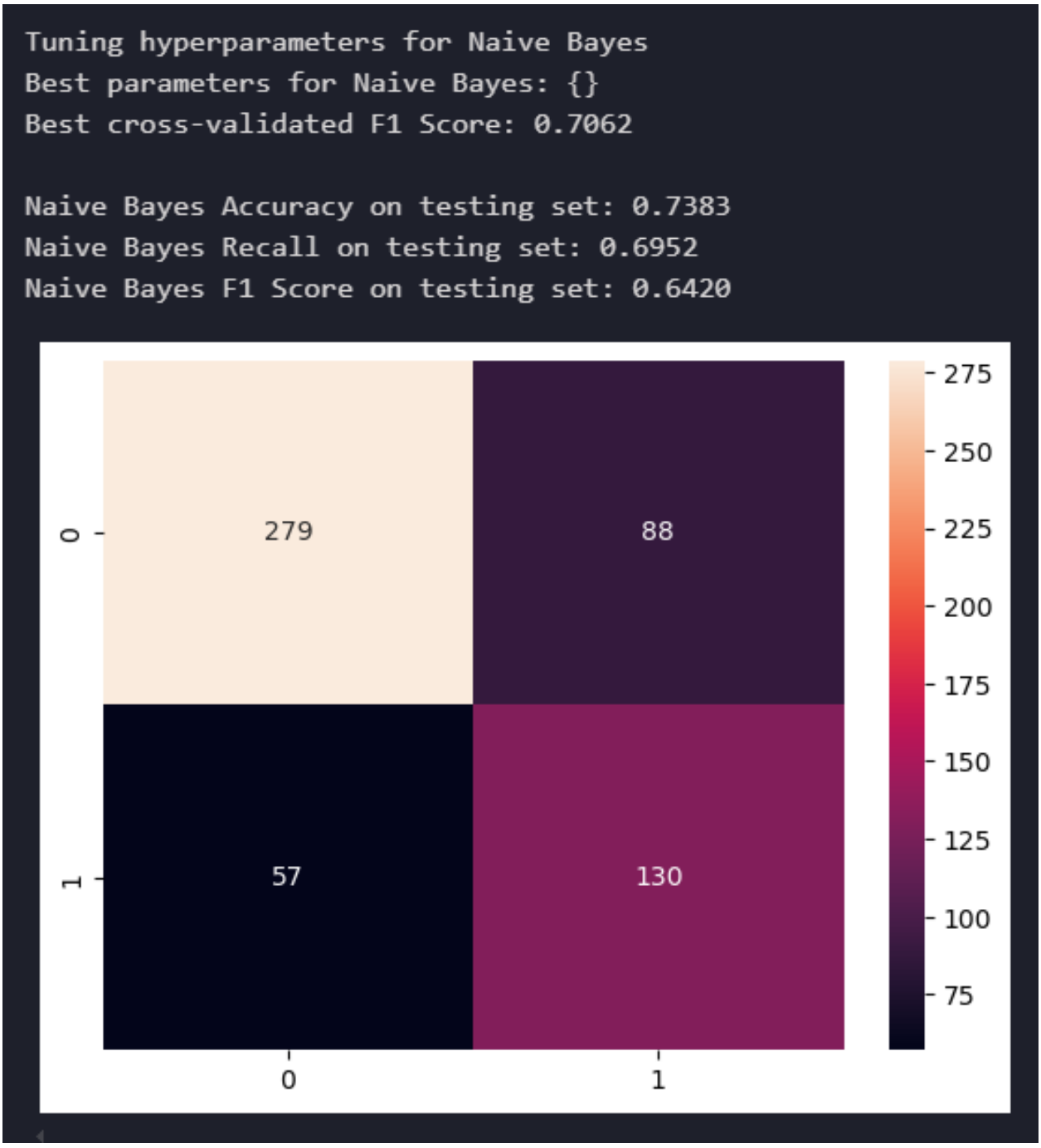


Figure 5.6: The cross-validated F1 Score, evaluation metrics, and confusion matrix of the Naïve Bayes model. There are no optimal parameters for this model as hyperparameter tuning was not performed on it. The reason for this is described in Appendix D.

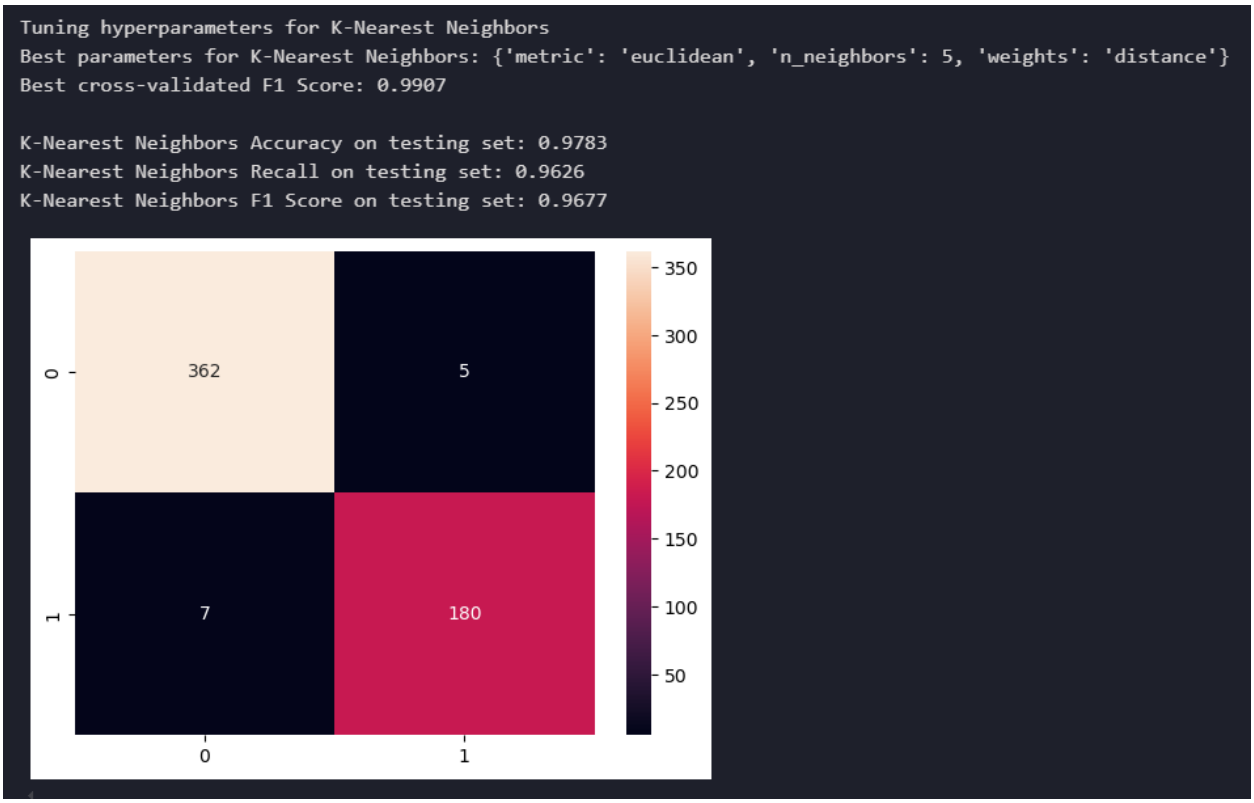


Figure 5.7: The optimal parameters, cross-validated F1 Score, evaluation metrics, and confusion matrix of the K-Nearest Neighbours model.

5.2.1 Overall summary of results

The introduction of stratified cross-validation and hyperparameter tuning[1] to the models resulted in significant improvements across almost all algorithms. Intriguingly, the Random Forest model was the only one to decrease in performance, with a decrease in all three metrics on both the training and testing data, leading to the KNN model outperforming it by a very slight margin (F1 score + 0.0026, Accuracy + 0.0017) due to one more false positive being identified in the Random Forest model. This suggests that the Random Forest's first iteration may have been overfitted. Despite this decrease, the model is still an excellent predictor.

Every model other than Random Forest and Naïve Bayes saw marked improvements in all three evaluation metrics, demonstrating the necessity of cross-validation and hyperparameter tuning in supervised learning tasks. Table 5.2 shows the improvements in each metric.

[1]Except Naïve Bayes, which had no parameters to tune (See Appendix D.)

Model	Accuracy	Recall	F1 Score
Random Forest	-0.35%	-1.6%	-0.49
SVM	+3.1%	+13.9%	+0.04
LR	+0.7%	+9.7%	+1.6
NB	-3%	+4.9%	-2.8
KNN	+2.5%	+4.5%	+3.6

Table 5.2: The differences in each evaluation metric in each second-iteration model on the unseen testing data.

Conclusion

6.1 Summary of results

This project aimed to develop and evaluate machine learning models for predicting the presence of diabetes mellitus using clinical data. Through extensive exploratory data analysis, data preprocessing, and model development, the following key results were achieved:

- Dataset Identification and Integration
  - The identification and integration of two datasets (Pima Indian and Frankfurt) resulted in a comprehensive dataset of 2,768 samples.
- Data Preprocessing and EDA
  - Assumptions and questions solved by exploratory data analysis revealed critical insights, including the presence of outliers, missing values, and class imbalance, which were addressed through constraining outliers to upper and lower quantiles, imputing missing data using KNN, and balancing classes with SMOTE.
- Model Development and Evaluation
  - K-Nearest Neighbours (KNN) emerged as the top-performing model after hyperparameter tuning, achieving 97.8% accuracy, 96.2% recall, and an F1-score of 96.6.
  - Random Forest performed exceptionally well, with 97.7% accuracy, 96.2% recall, and an F1-score of 96.5, only slightly behind KNN.
  - Support Vector Machine (SVM) showed significant improvement after tuning, reaching 85.7% accuracy, 85% recall, and an F1-score of 80.1.
  - Logistic Regression demonstrated moderate performance with 75% accuracy, 70.5% recall, and an F1-score of 65.6.
  - Naïve Bayes performed the least effectively, with 73.8% accuracy, 69.5% recall, and an F1-score of 64.2.

In conclusion, this project successfully developed machine learning models capable of accurately predicting diabetes, with the KNN and Random Forest models being most successful. These results demonstrate the potential of machine learning in supporting medical diagnosis, specifically for diabetes mellitus. However, further validation with larger, more complete datasets would be beneficial to enhance the models' robustness and generalizability in clinical settings.

6.2 Reflection on Individual Learning

Over the course of this project, I have enhanced my skills to a degree I previously thought impossible. I have thoroughly enjoyed expanding my Python skills for data science and machine learning with the use of industry standard packages like Pandas, NumPy and Scikit-Learn. This project in particular has heavily challenged me, requiring extensive research into many academic papers and web resources to further my knowledge of the medical field and machine learning trends. The most challenging part of this project was the use of GridSearchCV on five different models, though I eventually reached the solution of an iterative for-loop that ran the search on all five in a clean block of code.

Despite the accomplishments of this project, it is not without flaws. KNN imputation was also used on the testing set to address the missing data within it. While this did mean that more of the testing set could be used, this potentially comes at the expense of the data not being legitimate, as it was synthetically generated, and could have potentially caused overfitting.

I have always enjoyed writing code from as young as 11 years old. I originally began by creating games on the Roblox platform using their specialised fork of the Lua language, and expanded beyond that into Python as I furthered my knowledge of computing and IT. Also, I have a personal GitHub account, which I use for version control and iterative saving of my University work, including for this module in the CMP6202 repository, where every step taken in the production of this report and code is documented across many commits.

Irrespective of my personal enjoyment in writing code, the topic of this project was one that is close to my heart - members of my family suffer from type 2 diabetes and other health complications that arose from it, including heart attacks and strokes. Therefore, I am intensely interested in the applications of my passions in AI and Machine Learning to make a positive difference, and I believe that the knowledge I gained through this project will be a strong asset in doing so.

Bibliography

Adam, Sumaiya, Harold David McIntyre, Kit Ying Tsoi, Anil Kapur, Ronald C. Ma, Stephanie Dias, Pius Okong, Moshe Hod, Liona C. Poon, Graeme N. Smith, Lina Bergman, Esraa Algurjia, Patrick O'Brien, Virna P. Medina, Cynthia V. Maxwell, Lesley Regan, Mary L. Rosser, Bo Jacobsson, Mark A. Hanson, Sharleen L. O'Reilly, Fionnuala M. McAuliffe, and the FIGO Committee on the Impact of Pregnancy on Long-term Health and the FIGO Division of Maternal and Newborn Health (2023). "Pregnancy as an opportunity to prevent type 2 diabetes mellitus: FIGO Best Practice Advice". In: *International Journal of Gynecology & Obstetrics* 160 (S1), pp. 56–67. ISSN: 1879-3479. DOI: 10.1002/ijgo.14537.

Aftab, Shabib, Saad Alanazi, Munir Ahmad, Muhammad Adnan Khan, Areej Fatima, and Nouh Sabri Elmitwally (2021). "Cloud-Based Diabetes Decision Support System Using Machine Learning Fusion". In: *Computers, Materials & Continua* 68 (1), pp. 1341–1357. ISSN: 1546-2226. DOI: 10.32604/cmc.2021.016814.

Akmeşe, Ömer Faruk (Mar. 30, 2022). "Diagnosing Diabetes with Machine Learning Techiques". In: *Hittite Journal of Science and Engineering* 9 (1), pp. 9–18. ISSN: 2148-4171. DOI: 10.17350/HJSE19030000250.

AlZu'bi, Shadi, Mohammad Elbes, Ala Mughaid, Noor Bdair, Laith Abualigah, Agostino Forestiero, and Raed Abu Zitar (Feb. 2023). "Diabetes Monitoring System in Smart Health Cities Based on Big Data Intelligence". In: *Future Internet* 15 (2). Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, p. 85. ISSN: 1999-5903. DOI: 10.3390/fi15020085.

c3.ai (2024). *What is Ground Truth? | Machine Learning Glossary Definition*. C3 AI. URL: https://c3.ai/glossary/machine-learning/ground-truth/ (visited on 12/09/2024).

Dennison, Rebecca A., Eileen S. Chen, Madeline E. Green, Chloe Legard, Deeya Kotecha, George Farmer, Stephen J. Sharp, Rebecca J. Ward, Juliet A. Usher-Smith, and Simon J. Griffin (Jan. 2021). "The absolute and relative risk of type 2 diabetes after gestational diabetes: A systematic review and meta-analysis of 129 studies". In: *Diabetes Research and Clinical Practice* 171, p. 108625. ISSN: 01688227. DOI: 10.1016/j.diabres.2020.108625.

Dhanapalaratnam, Roshan, Tushar Issar, Leiao Leon Wang, Darren Tran, Ann M. Poynten, Kerry-Lee Milner, Natalie C.G. Kwai, and Arun V. Krishnan (Aug. 21, 2024). "Effect of Metformin on Peripheral Nerve Morphology in Type 2 Diabetes: A Cross-Sectional Observational Study". In: *Diabetes* 73 (11), pp. 1875–1882. ISSN: 0012-1797. DOI: 10.2337/db24-0365.

Diabetes UK (2024a). *How many people in the UK have diabetes?* Diabetes UK. URL: https://www.diabetes.org.uk/about-us/about-the-charity/our-strategy/statistics (visited on 11/27/2024).

Diabetes UK (2024b). *What causes type 2 diabetes?* Diabetes UK. URL: https://www.diabetes.org.uk/about-diabetes/type-2-diabetes/causes (visited on 12/14/2024).

Donnelly, Louise A., Rory J. McCrimmon, and Ewan R. Pearson (2024). "Trajectories of BMI before and after diagnosis of type 2 diabetes in a real-world population". In: *Diabetologia* 67 (10), pp. 2236–2245. ISSN: 0012-186X. DOI: 10.1007/s00125-024-06217-1.

Hayashi, Yoichi and Shonosuke Yukita (Jan. 1, 2016). "Rule extraction using Recursive-Rule extraction algorithm with J48graft combined with sampling selection techniques for the diagnosis of type 2 diabetes mellitus in the Pima Indian dataset". In: *Informatics in Medicine Unlocked* 2, pp. 92–104. ISSN: 2352-9148. DOI: 10.1016/j.imu.2016.02.001.

John DaSilva (2024). *Frankfurt Diabetes Dataset*. diabetes. URL: https://www.kaggle.com/datasets/johndasilva/diabetes (visited on 11/25/2024).

Joshi, Ram D. and Chandra K. Dhakal (July 9, 2021). "Predicting Type 2 Diabetes Using Logistic Regression and Machine Learning Approaches". In: *International Journal of Environmental Research and Public Health* 18 (14), p. 7346. ISSN: 1661-7827. DOI: 10.3390/ijerph18147346.

Khalili, Davood, Marjan Khayamzadeh, Karim Kohansal, Noushin Sadat Ahanchi, Mitra Hasheminia, Farzad Hadaegh, Maryam Tohidi, Fereidoun Azizi, and Ali Siamak Habibi-Moeini (Feb. 14, 2023). "Are HOMA-IR and HOMA-B good predictors for diabetes and pre-diabetes subtypes?" In: *BMC Endocrine Disorders* 23, p. 39. ISSN: 1472-6823. DOI: 10.1186/s12902-023-01291-9.

Khan, Moien Abdul Basith, Muhammad Jawad Hashim, Jeffrey Kwan King, Romona Devi Govender, Halla Mustafa, and Juma Al Kaabi (Mar. 2020). "Epidemiology of Type 2 Diabetes – Global Burden of Disease and Forecasted Trends". In: *Journal of Epidemiology and Global Health* 10 (1), pp. 107–111. ISSN: 2210-6006. DOI: 10.2991/jegh.k.191028.001.

Nelaj, Ergita, Margarita Gjata, Irida Kecaj, Ilir Gjermeni, and Mihal Tase (June 2023). "HIGH BLOOD PRESSURE IN THE NEWLY DIAGNOSED TYPE 2 DIABETES PATIENTS". In: *Journal of Hypertension* 41 (Suppl 3), e172. ISSN: 0263-6352. DOI: 10.1097/01.hjh.0000940640.80128.7a.

NHS (Oct. 16, 2022). *Peripheral neuropathy - Causes*. nhs.uk. Section: conditions. URL: https://www.nhs.uk/conditions/peripheral-neuropathy/causes/ (visited on 12/04/2024).

Ozbun, Andrew (July 6, 2021). *Properly Using SMOTE*. Nerd For Tech. URL: https://medium.com/nerd-for-tech/properly-using-smote-930924e81ab5 (visited on 12/17/2024).

Ruiz-Alejos, Andrea, Rodrigo M Carrillo-Larco, J Jaime Miranda, Robert H Gilman, Liam Smeeth, and Antonio Bernabé-Ortiz (Jan. 2020). "Skinfold thickness and the incidence of type 2 diabetes mellitus and hypertension: an analysis of the PERU MIGRANT study". In: *Public Health Nutrition* 23 (1), pp. 63–71. ISSN: 1368-9800. DOI: 10.1017/S1368980019001307.

Sivakumar, Muthuramalingam, Sudhaman Parthasarathy, and Thiyagarajan Padmapriya (Sept. 6, 2024). "Trade-off between training and testing ratio in machine learning for medical image processing". In: *PeerJ Computer Science* 10, e2245. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.2245.

Suastika, Ketut, Pande Dwipayana, Made Siswadi, and R.A. Tuty (Dec. 12, 2012). "Age is an Important Risk Factor for Type 2 Diabetes Mellitus and Cardiovascular Diseases". In: *Glucose Tolerance*. Ed. by Sureka Chackrewarthy. InTech. ISBN: 978-953-51-0891-7. DOI: 10.5772/52397.

Tahapary, Dicky Levenus, Livy Bonita Pratisthita, Nissha Audina Fitri, Cicilia Marcella, Syahidatul Wafa, Farid Kurniawan, Aulia Rizka, Tri Juli Edi Tarigan, Dante Saksono Harbuwono, Dyah Purnamasari, and Pradana Soewondo (Aug. 1, 2022). "Challenges in the diagnosis of insulin resistance: Focusing on the role of HOMA-IR and Tryglyceride/glucose index". In: *Diabetes & Metabolic Syndrome: Clinical Research & Reviews* 16 (8), p. 102581. ISSN: 1871-4021. DOI: 10.1016/j.dsx.2022.102581.

TrainInData (Mar. 28, 2023). *Overcoming Class Imbalance with SMOTE: How to Tackle Imbalanced Datasets in Machine Learning*. Train in Data's Blog. URL: https://www.blog.trainindata.com/overcoming-class-imbalance-with-smote/ (visited on 12/17/2024).

TrainInData (July 24, 2024). *KNN imputation of missing values in machine learning*. Train in Data's Blog. URL: https://www.blog.trainindata.com/knn-imputation-of-missing-values-in-machine-learning/ (visited on 12/17/2024).

UCI Machine Learning (2024). *Pima Indians Diabetes Database*. Pima Indians Diabetes Database. URL: https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database (visited on 11/25/2024).

Zou, Qiong, Yang Zhang, and Chang Sheng Chen (Feb. 13, 2024). "Construction and Application of a Machine Learning Prediction Model Based on Unbalanced Diabetes Data Fusion". In: *Proceedings of the 2023 International Joint Conference on Robotics and Artificial Intelligence*. JCRAI '23. New York, NY, USA: Association for Computing Machinery, pp. 114–123. ISBN: 979-8-4007-0770-4. DOI: 10.1145/3632971.3633348.