Developing a classification model for linear B-Cell epitopes

Table of Contents

Abstract

This report details the process of building a solution to predict different classes from a training dataset containing 15,000 labelled records. Data exploration revealed that most records had features with missing entries. Features were found to be correlated with others and no singular feature could determine the class of a record. Data pre-processing was completed to address the missing records by either column or row deletion, or predicting the missing values, records were scaled also for models such as KNN. Different feature selection techniques were tested to pick the best 15 features. Oversampling was then done to address the class imbalance. Finally, Model tuning was completed to produce the best f1 score for each model. In the end, the KNN model was used on the unclassified dataset.

Introduction

Experimental determination of biological properties of protein sequences remains an expensive and time-consuming bottleneck in the development of new vaccines and serological tests for infectious diseases. A dataset was provided of observations related to linear B-Cell epitopes which are protein regions of interest for a variety of applications in immunology. The goal of this report is to develop a classification model to predict the class of new, previously unseen observations with the highest possible f1 score as well as document the process of developing the model.

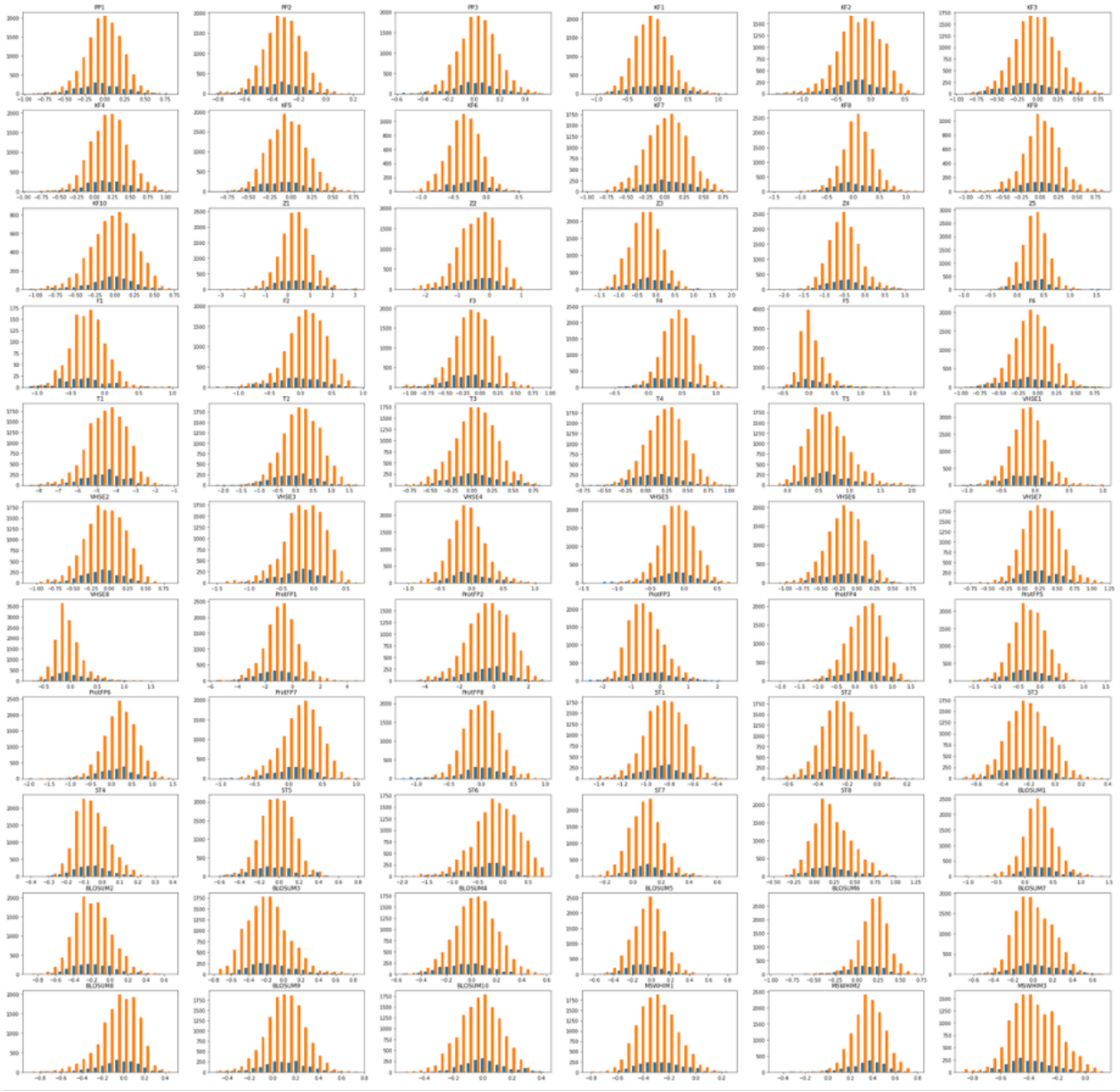
Data Exploration

Jupyter Notebook is a Web-based application that allows a document to be created with live-code, narration, and visualisation. Using the programming language Python, I was able to organise the data set into a Dataframe to understand and visualise its different properties through text, tables, and graphs.

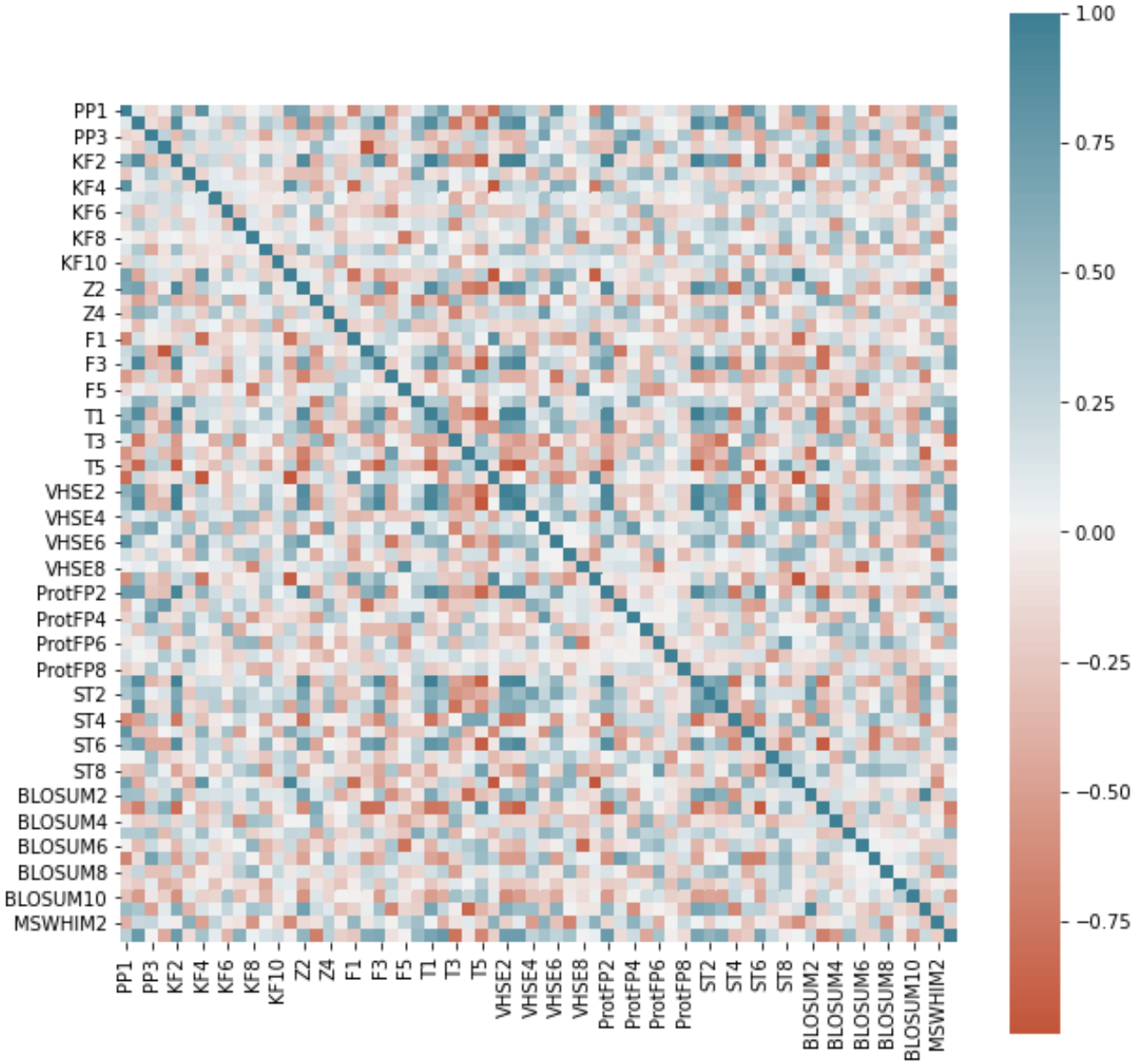
Initial exploration

- There are 15,000 uniquely identified records.
- There are 67 input attributes.
- There are 6 input attributes with missing data:
  1. KF6: 7693 (51%)
  2. KF9: 7693 (51%)
  3. KF10: 7693 (51%)
  4. F1: 13814 (92%)
  5. T2: 496 (3.3%)
  6. ST7: 81 (0.54%)
- The data set is imbalanced as a total of 13046 (86.97%) “Negative” results and 1954 “Positive” (13.03%) Results for the “Class” Attribute.
- When a histogram of the “Positive” and “Negative” instances of a feature is displayed on the same plot, no single feature seems to be able to classify the data.
- The data distributions of most of the classes of features appear Gaussian, only some appeared to be skewed and non-Gaussian.
- Many features are correlated with each other, including ones which have missing data.

Feature histograms



Feature correlation



Data Pre-processing

Missing data

Missing Data Mechanisms

In data exploration there were 6 features input attributes which contained missing values. To determine the best approach to dealing with it, we must understand the assumptions based on the reasons for the missing data. This can be split into three types of missing data: missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR). MCAR is defined as when the probability that the data missing is not related to either the specific value which is supposed to be obtained or the set of observed responses. Data that is MCAR allows analysis to remain unbiased. MAR is the more realistic assumption; it is probability that the responses are missing depends on the set of observed responses. MNAR is when data does not fall into the categories of MCAR and MAR. To obtain unbiased estimates, the missing values can be modelled [1].

Attribute deletion

The first feature to explore is F1, 92% of its values are missing which means using methods to remove rows containing missing values would sacrifice a large amount of useful data, and even though there was a high correlation between F1 and three other features, predicting the missing values would create too much bias. There are no reasons available for the data being missing, so it can be assumed that it is MCAR. The approach I took was to drop the attribute entirely, I followed the same approach for KF6, KF9, and KF10. Attributes KF6, KF9, and KF10 had the same percentage of missing data at 51%, a row either had no data of these features missing or had all data from these features missing. The dataset was created by parsing existing proteomic sequences, retrieved, and consolidated from the online databases IEDB and GenBank so a theory is that one or more sources did not contain these features. Although this can be defined as MNAR, replacing the data with an estimation would result in a large bias.

Listwise deletion

The assumption for the missing data in attribute ST7 is MCAR. With the missing data only accounting for 0.54%, the deletion of rows that contain a missing value in that column were deleted. This resulted in 0.56% of positive classes being deleted and 0.54% of negative classes being deleted.

Regression imputation

The assumption for the missing data in attribute T2 is MCAR. Although 3.3% of data was missing in the T2 column entry, deleting those rows might not have a major impact on training the model, but it is possible that these rows could contain useful data. Computing the pairwise correlation between T2 and the remaining attributes, I discovered that T2 correlated highly with several of them. With this knowledge I decided to get estimates for the missing data by modelling using linear regression. The five following attributes were picked: ProtFP2, MSWHIM3, ST6, BLOSUM3, and PP1, which has a correlation of 0.805, 0.779, 0.728, -0.727, and 0.702 respectively with T2. A correlation of 1 indicates a perfect correlation, 0 indicated no correlation and -1 indicates a perfect inverse correlation. Using rows with no missing data to train and test the model, a R2 score of 0.901 was achieved, this shows that the predicting the missing data can be very accurate [2].

Scaling

Scaling is important as some algorithms focus on the magnitude of features while neglecting the units. Features with high magnitudes will have a larger influence than features with low magnitudes. In finding the best model for the data, I will be experimenting with k-nearest neighbours, for this reason I will be using the min-max scalar which scales the values of features between 0 and 1 [3].

Feature Selection

Benchmark models

After pre-processing 5 different classification models were chosen. Benchmark models were created to discover if feature selection and parameter tuning made improvements. K-fold cross validation was done on the data set to get 8 variations of training and testing data. The benchmark models were trained and tested with each variation and the variation that produced the best F1 score for each model was selected. The F1 score and accuracy for the selected variations was saved. As accuracy only accounts for the true negatives and positives, F1 score is preferred as it also accounts for false positives and true negatives. Also in this classification problem, imbalanced class distribution exists and thus F1-score is a better metric to evaluate the model on [11].

Model	F1 Score	Accuracy
Gaussian Naive Bayes	0.399	79.4%
Decision Tree	0.558	88.7%
MLP Classifier	0.498	89.3%
Logistic Regression	0.191	88.7%
k-Neighbours Classifier	0.608	90.4%

Variance

The first attempt at feature selection was to choose the 15 features with the highest variations, this proved to produce inferior results to the benchmark models so that method was abandoned.

Model	F1 Score (15 features)	Accuracy
Gaussian Naive Bayes	0.206	80.9%
Decision Tree	0.519	88.1%
MLP Classifier	0.382	88.7%
Logistic Regression	0.113	88.4%
K Neighbours Classifier	0.570	89.7%

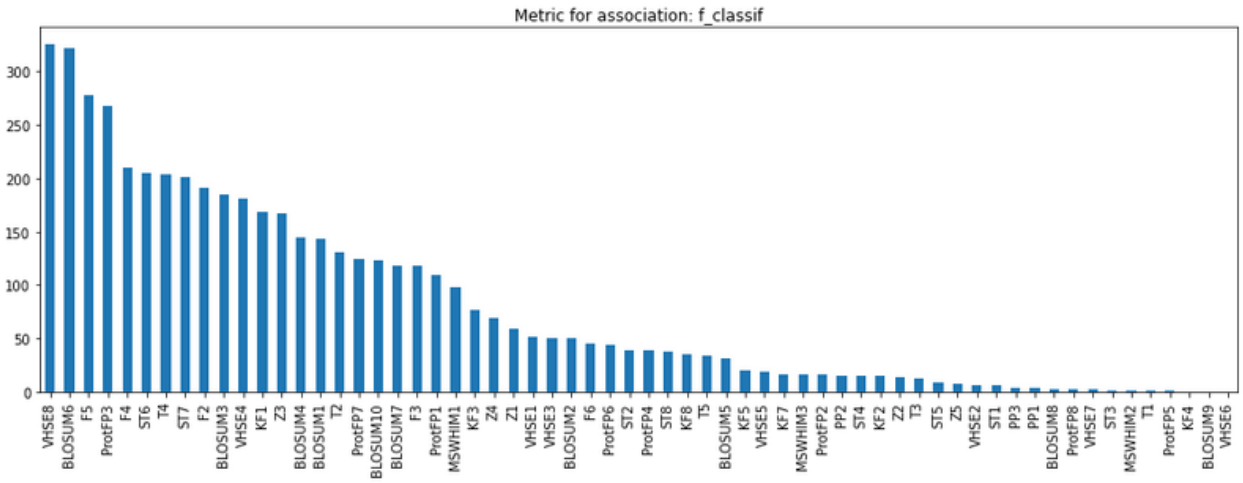
Correlation

In data exploration I discovered that multiple features were highly correlated with each other. Removing one of two highly correlated features could have a small effect on models' performance and improve feature selection methods. Therefore, I tried the following feature selection methods on the full pre-processed dataset as well as the dataset with one of two highly correlated features removed. A feature chosen to be removed must be at least 80% correlated with another feature and have a lower score than that other feature in the feature selection method.

Univariate measures

My seconds attempt was to select the best features based on univariate statistical tests. This selects variables most related to the target outcome. The metrics for association I choose were f_classif and chi2. F_classif is based on the analysis of Variance (ANOVA) statistical test. Chi2 performs the chi-squared statistic for categorical targets. The higher a score is the more related to the target outcome it is [4]. Additionally, the k best features were tested on the benchmark models, with k increasing from 1 to 15.

F_classif



Feature selection without correlation

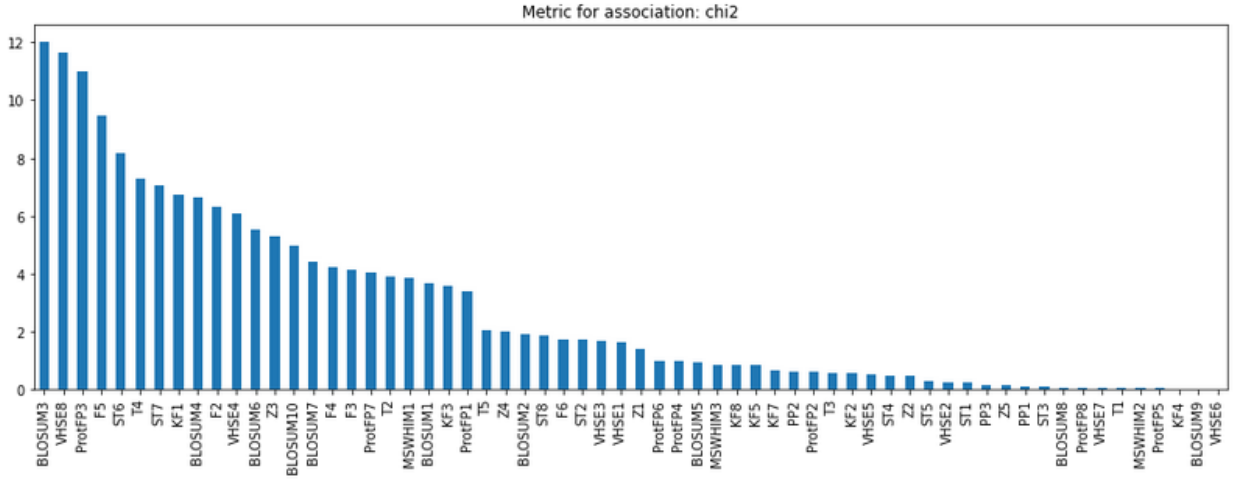
Model	F1 Score (14 features)	Accuracy
Gaussian Naive Bayes	0.376	83.1%
Decision Tree	0.519	87.8%
MLP Classifier	0.370	88.6%
Logistic Regression	0.124	88.6%
K Neighbours Classifier	0.564	89.4%

Feature selection with correlation

Model	F1 Score (15 features)	Accuracy
Gaussian Naive Bayes	0.423	84.3%

Decision Tree	0.539	88.2%
MLP Classifier	0.369	88.5%
Logistic Regression	0.132	88.5%
K Neighbours Classifier	0.599	90.2%

Chi2



Feature selection without correlation

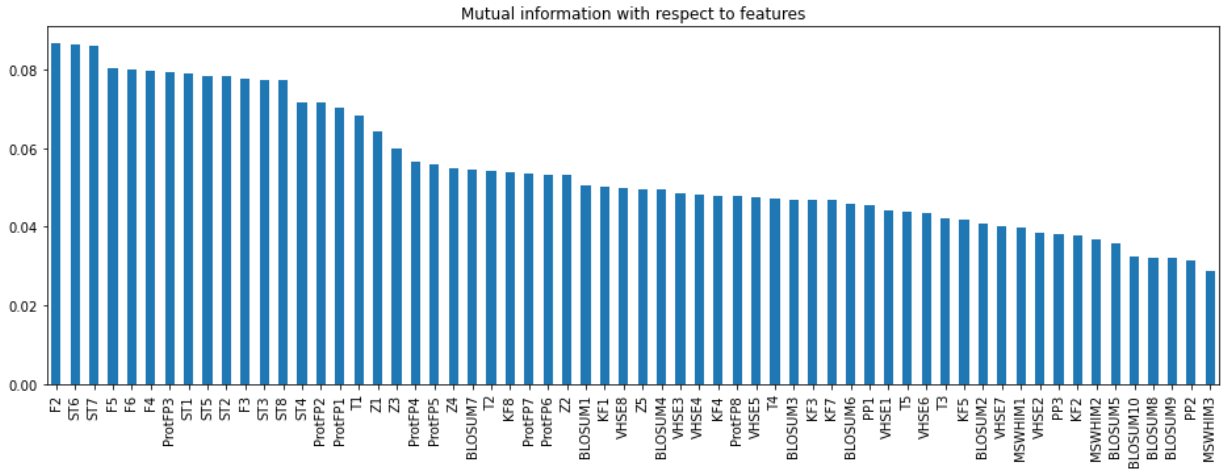
Model	F1 Score (14 features)	Accuracy
Gaussian Naive Bayes	0.396	82.8
Decision Tree	0.527	88.3
MLP Classifier	0.322	88.4
Logistic Regression	0.175	88.6
K Neighbours Classifier	0.548	88.7

Feature selection with correlation

Model	F1 Score (15 features)	Accuracy
Gaussian Naive Bayes	0.440	85.3%
Decision Tree	0.526	88.3%
MLP Classifier	0.345	88.5%
Logistic Regression	0.177	88.8%
K Neighbours Classifier	0.614	90.6%

Mutual information gain

My third attempt was to use mutual_info_classif. It measures the mutual information between a matrix containing a set of feature vectors and the target. It provides an understanding in how good of a predictor is a feature is. All features retrieve a score, the higher the score the better the predictor a feature is [5].



Feature selection without correlation

Model	F1 Score (14 features)	Accuracy
Gaussian Naive Bayes	0.387	83.7%
Decision Tree	0.540	88.5%
MLP Classifier	0.349	88.5%
Logistic Regression	0.115	88.6%
K Neighbours Classifier	0.548	89.3%

Feature selection via correlation

Model	F1 Score (15 features)	Accuracy
Gaussian Naive Bayes	0.343	82.5%
Decision Tree	0.523	88.0%
MLP Classifier	0.332	88.1%
Logistic Regression	0.095	88.5%
K Neighbours Classifier	0.571	90.2%

Selected Features

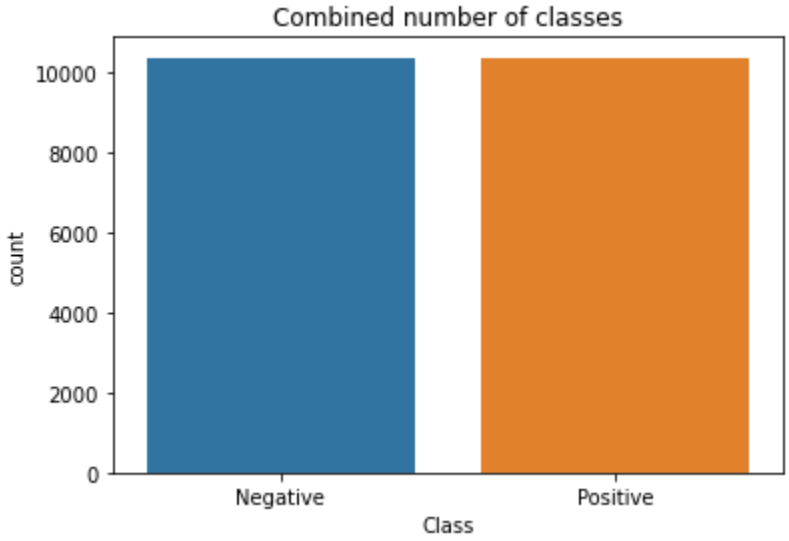
The best results from feature selection came from using f_chi2 on a dataset that had correlated features removed. For most models, removing features produced a lower F1 score, but accuracy was either slightly better or worse. The chosen features were BLOSUM4, KF3, BLOSUM7, VHSE4, ProtFP7, BLOSUM1, Z3, VHSE8, ST7, T4, F4, T2, BLOSUM3, BLOSUM10, and KF1.

	Benchmark		After feature selection	
Model	F1 Score	Accuracy	F1 Score	Accuracy
Gaussian Naive Bayes	0.399	79.4%	0.440	85.3%
Decision Tree	0.558	88.7%	0.526	88.3%
MLP Classifier	0.498	89.3%	0.345	88.5%
Logistic Regression	0.191	88.7%	0.177	88.8%

K Neighbours Classifier	0.608	90.4%	0.614	90.6%

Oversampling

With only 13% of the data belonging to the "Positive" class, there is a huge under-representation of the data. Synthetic Minority Oversampling Technique (SMOTE) is a type of data augmentation for the minority class. Feature selection was performed before oversampling as most variable selection methods assume that the samples are independent [10]. Instead of duplicating examples from the minority class, it creates new examples that are synthesized from the existing minority class.



New benchmark models

The same method from creating the first benchmark was used to create the new one with oversampled data. F1 scores improved for all models, but the accuracy did decrease for some. These new benchmark models will be used to compare tuned models.

Model	F1 Score	Accuracy
Gaussian Naive Bayes	0.593	63.9%
Decision Tree	0.899	90.1%
MLP Classifier	0.837	83.8%
Logistic Regression	0.673	68.3%
K Neighbours Classifier	0.926	92.3%

Model Tuning

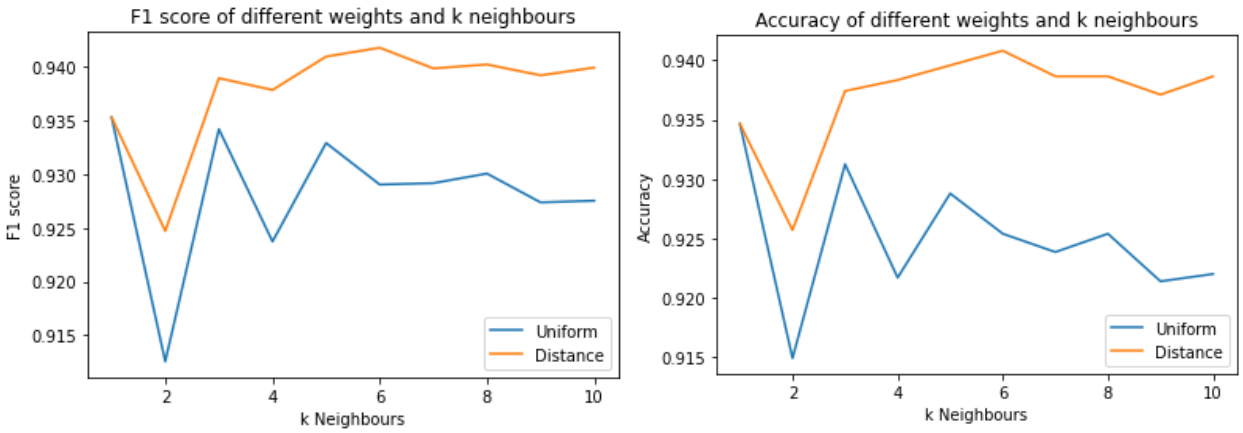
After data processing and feature selection, different parameters of each model will be tuned to see if a better result can be obtained.

k-nearest Neighbour

The k-nearest neighbours (KNN) are a non-parametric, lazing learning method. This means that no assumptions are made about the data and there is not explicitly any training done. When a test sample is used on the classifier model, it compares the sample to the closest neighbours around it [6]. Two parameters can be tested, the number of neighbours and the weight. Two different weights to be used are uniform and distance weights. Uniform weights are where all points in each neighbourhood are weighted equally while with distance, the points closest to the sample have a greater influence then neighbours further away.

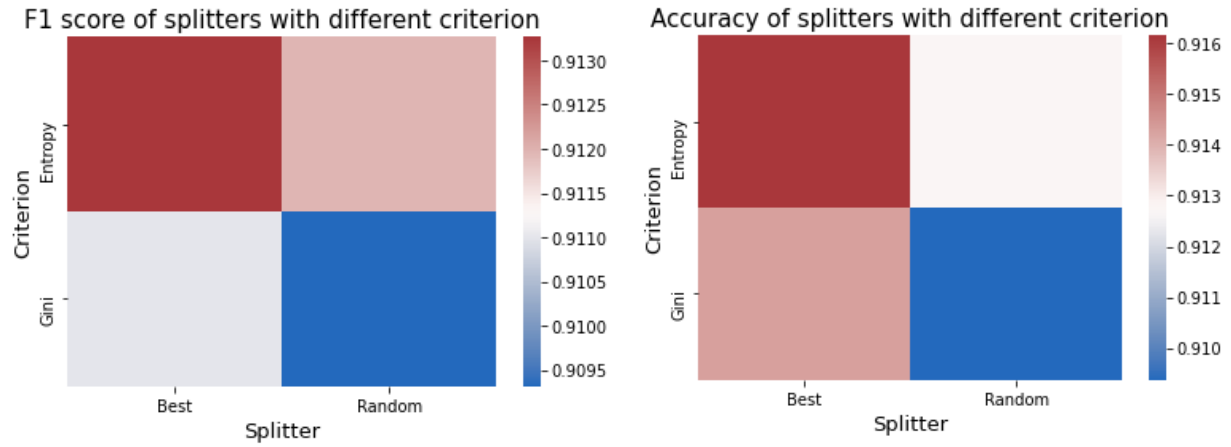
Using the oversampled dataset, I performed an 8 folds cross-validation method for k values from 1 to 10 to find the best results. For the distance weight, the best KNN model had 6 neighbours, it produced the highest F1 score of 0.942 and accuracy of 94.1%. For the uniform weight, the best KNN model had 3 neighbours, it also produced an F1 score of 0.934 and an accuracy of 93.1%. For this reason, the best KKN model was distanced weighted and had 6 neighbours.

Distance

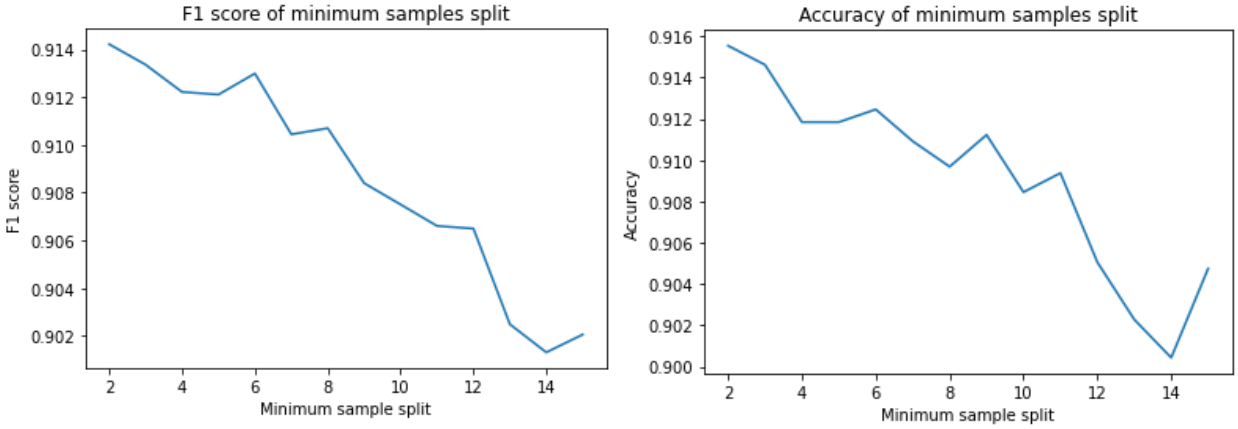


Decision tree

Decision trees are non-parametric supervision learning method. The method creates a model that predicts the value of a target variable by learning simple decision rules inferred from data features. To find the best configuration of decision trees, I will be testing combinations of the criterion and splitter, as well as the minimum number of samples require to split an internal node. The criterion is the function to measure the quality of a split, supported criteria include ‘gini’ for Gini impurity and ‘entropy’ for information gain [7]. The splitter is the strategy used to choose the split at each node. The two splitter strategies include ‘best’ which is used to choose the best split, and ‘random’ which is used to choose the best random split. The best combination of criterion and splitter found was ‘entropy’ and ‘best’ which had a higher F1 score and accuracy then the other combinations.



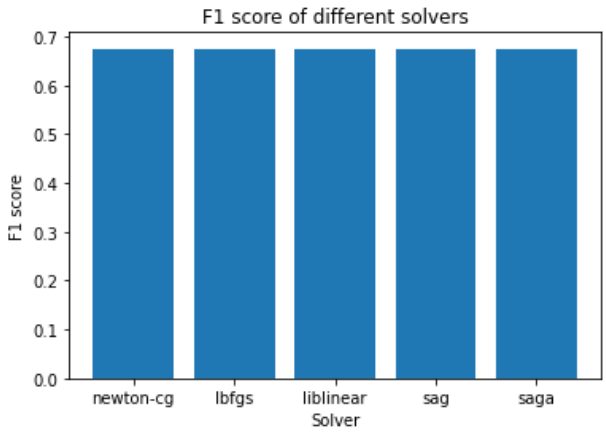
For the minimum number of samples required to split an internal node, the best combination of criterion and splitter was used with test samples ranging from 2 to 15. The optimal minimum sample split was 2 as it had the highest F1 score and accuracy which were 91.6% and 0.914, respectively.

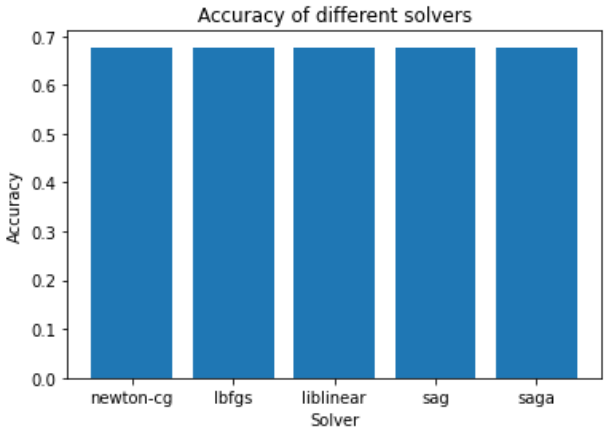


Logistic Regression

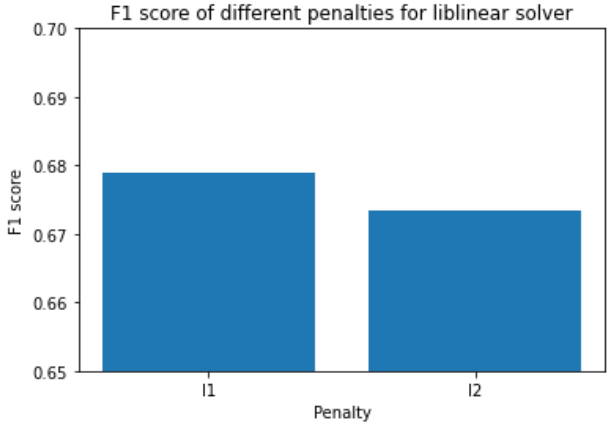
The first Logistic regression parameter I experimented with was the solver which is an algorithm to use in the optimization problem. The five solvers used were ‘newton-cg’, ‘lbfgs’, ‘liblinear’, ‘sag’ and ‘saga’. All solvers produced the same results in terms of accuracy and F1 score. Using the ‘liblinear’ solver I wanted to find the best penalty to use, this is used to specify the norm used in the penalization [8]. The ‘l1’ penalty was the best option as it produced the highest F1 score and accuracy which were 0.679 and 68.0% respectively.

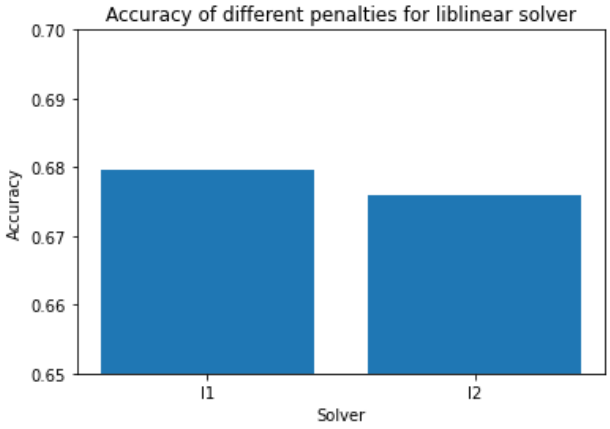
Solver





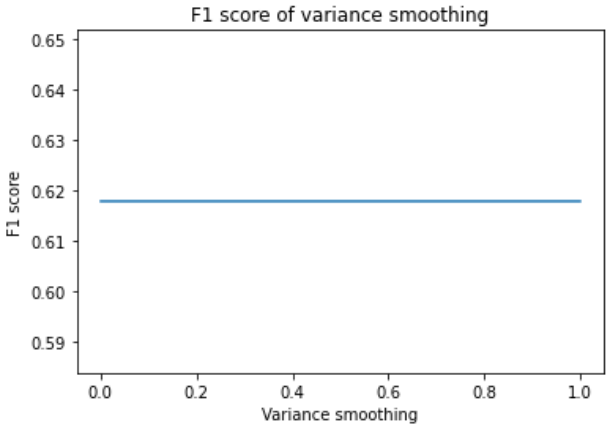
Penalization

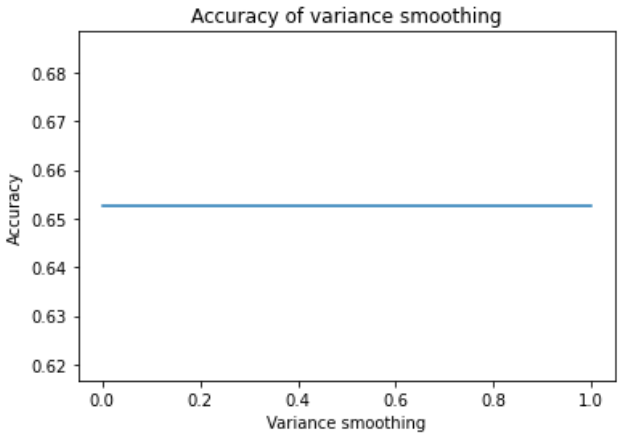




Gaussian Naive Bayes

The Gaussian Naïve Bayes had only 2 model parameters: priors and variance smoothing. For the dataset I only tested a variance smoothing from 0 to 1. Variance smoothing is the portion of the largest variance of all features that is added to variances for calculation stability [9]. Changing variance smoothing had no effect on the model as the F1 score remained 0.618, and the accuracy remained 65.3%.

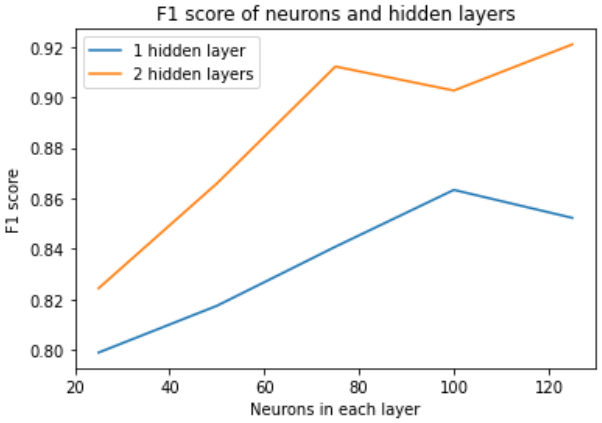


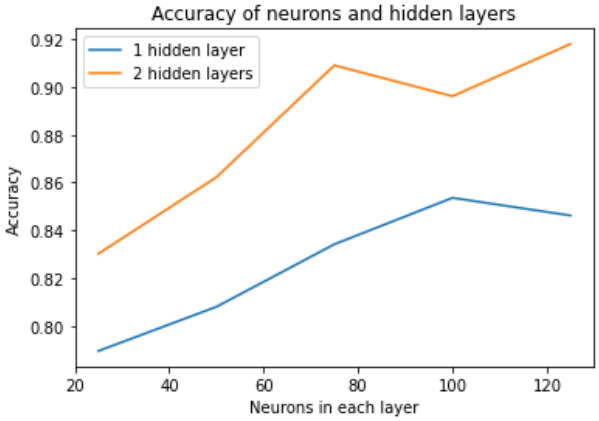


Multi-layer Perceptron classifier

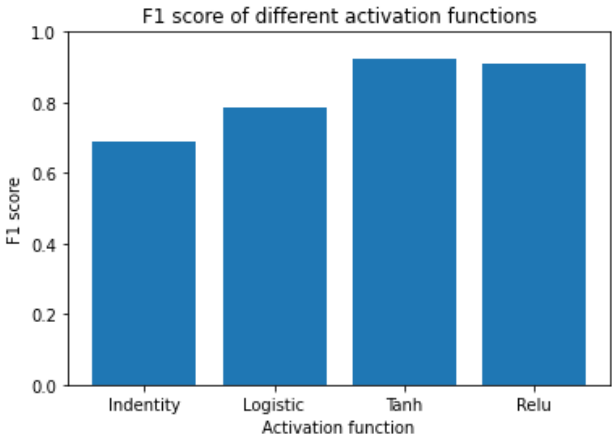
A large part of neural networks is the hidden layer sizes and number of neurons in each layer. I experimented with a range of neurons from 25 to 125 in intervals of 25 for one and two hidden layers. For two hidden layers in a test, each layer contained the same number of neutrons. I noticed that going from 1 to 2 layers, the F1 score increased a lot. The best option was 2 hidden layers with 75 neurons in each of them, this produced a high f1 score of 0.925 and the highest accuracy 92.1%. Using the best number of hidden layers and neurons, I tested different activation functions, the best activation function was the 'tahn'.

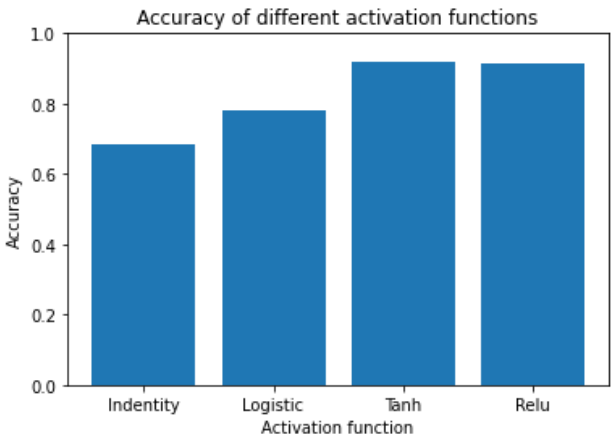
Hidden layers and neurons





Activation functions





Results

Model	Description	Accuracy		F1 score	
		Benchmark	Final	Benchmark	Final
Gaussian Naive Bayes	Default	65.3%	65.3%	0.618	0.618
Decision Tree	Criterion: entropy Min samples splits: 2	91.6%	91.6%	0.899	0.914
MLP Classifier	Maximum iterations: 1000 Hidden layer sizes: (75,75,75) Activation function: tahn	83.8%	92.1%	0.837	0.925
Logistic Regression	Maximum iterations: 1000 Penalty: l1 Solver: liblinear	68.3%	68.0%	0.673	0. 679
k Neighbours Classifier	Weights: distance N_neighbors: 2	92.3%	94.1%	0.926	0.942

Conclusion

In data exploration it was discovered that the features correlated, this was useful in predicting accurate missing data. Reducing the dataset to 15 features allowed the exploration and discovery of different features selection methods. While researching how to deal with imbalanced classes, oversampling proved to be a very useful tool. After tuning all the models, the k Nearest Neighbours classifier produced the best results as it had the highest accuracy, and most importantly, the highest F1 score. For this reason, that model was used on the unclassified dataset. On the unclassified dataset, out of the 15 columns preselected, ST7 an T2 had missing entries. For T2 the missing values where predicted and for ST7 the missing values were replaced with the mean of the column. Lastly, the dataset was scaled and had classes predicted for each record and stored in a CSV file.

References

[1] H. Kang, "The prevention and handling of the missing data", *National Institudes of Health*, 2013. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3668100/. [Accessed: 10- Apr- 2021].

[2] S. Kumar, "Predict Missing Values in the Dataset", *Medium*, 2020. [Online]. Available: https://towardsdatascience.com/predict-missing-values-in-the-dataset-897912a54b7b. [Accessed: 10- Apr- 2021].

[3] S. Asaithambi, "Why, How and When to Scale your Features", *Medium*, 2017. [Online]. Available: https://medium.com/greyatom/why-how-and-when-to-scale-your-features-4b30ab09db5e. [Accessed: 10- Apr- 2021].

[4] J. Mueller and L. Massaron, "How to Use Python to Select the Right Variables for Data Science - dummies", *dummies*. [Online]. Available: https://www.dummies.com/programming/big-data/data-science/how-to-use-python-to-select-the-right-variables-for-data-science/. [Accessed: 11- Apr- 2021].

[5] "What are Mutual_info_regression and Mutual_info_classif used for?", *Stack Overflow*, 2020. [Online]. Available: https://stackoverflow.com/questions/61708970/what-are-mutual-info-regression-and-mutual-info-classif-used-for. [Accessed: 11- Apr- 2021].

[6] T. Lin, "Day 3 — K-Nearest Neighbors and Bias–Variance Tradeoff", *Medium*, 2018. [Online]. Available: https://medium.com/30-days-of-machine-learning/day-3-k-nearest-neighbors-and-bias-variance-tradeoff-75f84d515bdb. [Accessed: 12- Apr- 2021].

[7] "1.10. Decision Trees", *Scikit-learn.org*, 2021. [Online]. Available: https://scikit-learn.org/stable/modules/tree.html. [Accessed: 12- Apr- 2021].

[8] "sklearn.linear_model.LogisticRegression", *Scikit-learn.org*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. [Accessed: 13- Apr- 2021].

[9] "sklearn.naive_bayes.GaussianNB", *Scikit-learn.org*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html. [Accessed: 13- Apr- 2021].

[10] R. Blagus and L. Lusa, "SMOTE for high-dimensional class-imbalanced data", *BMC Bioinformatics*, vol. 14, no. 1, 2013. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3648438/. [Accessed 12 April 2021].

[11] P. Huilgol, "Accuracy vs. F1-Score", *Medium*, 2019. [Online]. Available: https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2. [Accessed: 11- Apr- 2021].