Irish Planning Permission Approval Predictions

Bríd O'Donnell - 17330488
Juan Montenegro - 22334217

Introduction

This project "Irish Planning Application Approval Predictions" aims to accurately predict whether a planning permission will be approved or rejected by a local Irish housing authority. The motivation for this project is the ongoing housing crisis within Ireland and the current lack of housing supply. This supply is constrained by the planning system within Ireland and therefore this project hopes to illuminate this system. Planning permission within Ireland is administered by local council authorities, of which there are 31. While there are other overarching authorities including the judicial courts and Department of Housing, the vast majority of developments begin and end at the local level through their application process. In this project, a prediction model was developed to allow users to identify the most likely outcome of their planning application. The input to our algorithm is a series of properties related to a planning application. We then use a linear regression classifier and KNN classifier to output a predicted decision regarding the application This will hopefully allow the users to better tailor their applications and development plans in order to avoid rejection. This will save time and money and encourage more developments in the future, alleviating the housing crisis.

Dataset and Features

The primary datasets for this project was sourced from the Department of Housing, Local Government and Heritage as a part of their open data policy and can be found on the following website: https://data-housinggovie.opendata.arcgis.com/. The key dataset used is the *Planning Application Points*, which described as "the merged Planning Registers of participating Irish Local Authorities and includes all Planning Applications received since 2012"[1]. Given the long timeframe of this merged dataset, the dataset has over 400,000 data points and is regularly updated. Please note that the dataset was downloaded on November 6. Given the size of the dataset, we filtered the dataset to applications that have received a decision and were received between 1st January 2017 and 1st October 2022.

The dataset contains 39 columns, many of which were dropped as they were considered relevant in the application decision, such as AppealStatus and ExpiryDate. The columns remaining following this drop included the Latitude and Longitude values, the Planning Authority, Application Type, Area of Site, Number of Residence Units, Whether the development is a one-off house, the floor area of the development, the date the application was received and finally the decision on the application.

[1] Planning application points (2022) Dept of Housing, Local Government and Heritage Open Data. Dept of Housing, Local Government and Heritage. Available at: https://data-housinggovie.opendata.arcgis.com/datasets/housinggovie::planning-application-points/explore?location=53.380267%2C-8.304558%2C8.38 (Accessed: November 6, 2022).

Because of the multi-source nature of the dataset, there was a large amount of missing data and inconsistent classification, requiring significant cleaning. Null values were removed or in the case of *OneOffHouse* were substituted with dummy values. The decision values that couldn't be classified as either Granted or Refused were removed from the dataset, though these cases only comprised less than 1,000 of the dataset's rows. Regardless, the process of standardizing the decision column values was a laborious task. Following this, the decision column represented a granted application as having the value of 1, while a refused application had a value of 0.

Additional shapefiles available on https://data-housinggovie.opendata.arcgis.com/ were also combined with the dataset. These shapefiles included *Cities and Suburbs, Key Growth Centres* and *Metropolitan Areas,* which provided boundary data for recognised areas within Ireland that have already been significantly built up or the government hopes to develop. Using the latitude and longitude values from the dataset, the area in which the site is located was identified and this data was added to the dataset. This process was completed through the sf library in R. R was used instead of Python, as one of the team members had specific experience completing this task through R.

Since many of the variables were categorical, they were converted into numeric representation using the get_dummies function in order to perform a decision tree classifier. Finally, given the imbalance of the dataset regarding the decision variable, SMOTE (Synthetic Minority Over-sampling Technique) was used to compensate. SMOTE oversamples the minority class by creating 'synthetic' examples rather than copies. It involves some methods, including nearest neighbors, to generate plausible new examples. The final dataset consisted of 68316 data points, with a breakdown of the decision variables shown in Table 1.
We also use decision trees for feature importance, using the values of X and Y, and obtain the best features that impact our models.

Feature Importance
Feature importance is calculated as the decrease in node impurity weighted by the probability of reaching that node. The node probability can be calculated by the number of samples that reach the node, divided by the total number of samples. The higher the value the more important the feature. The importance of a feature is computed as the (normalized) total reduction of the criterion brought by that feature. It is also known as the Gini importance. Once the most important features have been obtained, we proceed to eliminate those that were not so relevant to train our model.

	Before SMOTE		**After SMOTE**	
Decision	**Size**	**Percentage**	**Size**	**Percentage**
1 (Granted)	60293	88.26%	48205	50%
0 (Refused)	8023	11.74%	48205	50%

Total	68316		96410	

Table 1: Breakdown of the decision

Methods

We tested three algorithms used when creating this prediction model: Decision Tree Classifier, KNN Classifier and Logistic Regression. Explanations for how these algorithms work are explained below. Please note that the Decision Tree Classifier was used to select the best features as opposed to creating a prediction model.

DecisionTreeClassifier

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

Logistic Regression

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes. In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no). Mathematically, a logistic regression model predicts P(Y=1) as a function of X.

KNNClassifier

The k-nearest neighbor algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

Experiments/Results/Discussion

As explained above, two different approaches were taken to create a classifier model: a Logistic Regression Classifier and a KNN Classifier. To train the models, we use the SMOTE data for the reason that it is balanced. In this section, the implementation (including the hyperparameter tuning) of the logistic regression classifier is discussed, followed by the implementation of the KNN Classifier will be covered. Following this, the models are evaluated and the results are discussed, the final outcome being a comparison between the two models.

Logistic Regression Implementation

For the hyperparameter tuning of the logistic regression classifier, `np.geomspace(1e-5, 1e5, num=20)` was used to generate a range of 20 values for C, starting from 0.048329 to 100000.000000. Cross validation is then used to identify the most optimal c value from a range of values using an errorbar plot. As seen in Figure 1, a number of plots were created with different lengths of x axis, to aid the visual identification of the optimal value._Based on the plots as well as the directly viewing the mean errors for each C value, the two most optimal C values were C=0.048329 and C=0.000113. However, the logarithmic loss or log loss value for the optimal C value, which indicates how close the prediction probability is to the corresponding actual/true value, was also calculated and since C=0.048329 had the smaller log loss value at 0.681287, it was chosen for our final model.

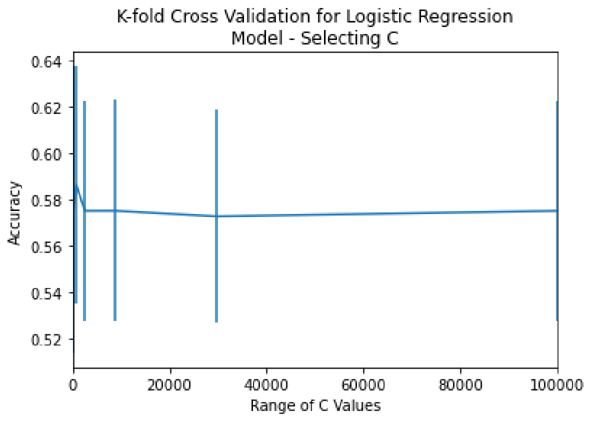

Figure 1a: Errorbar for Logistic Regression (Broad C Value Range)

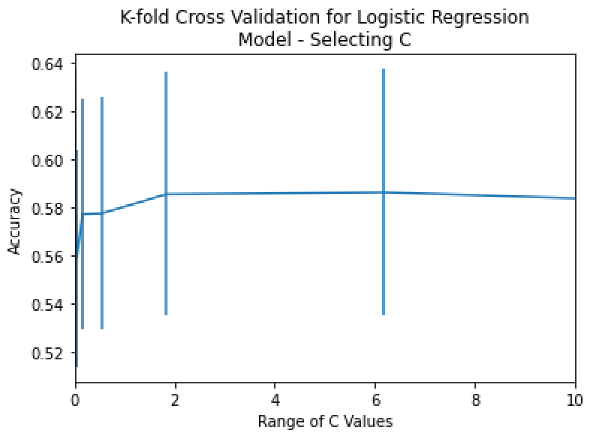

Figure 1b: Errorbar for Logistic Regression (Narrower C Value Range)

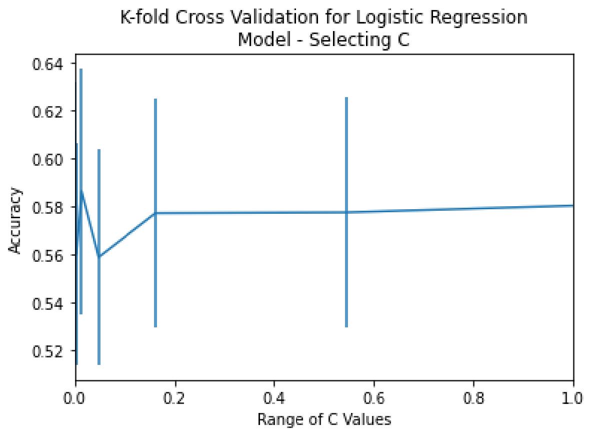

Figure 1c: Errorbar for Logistic Regression (Narrower C Value Range)

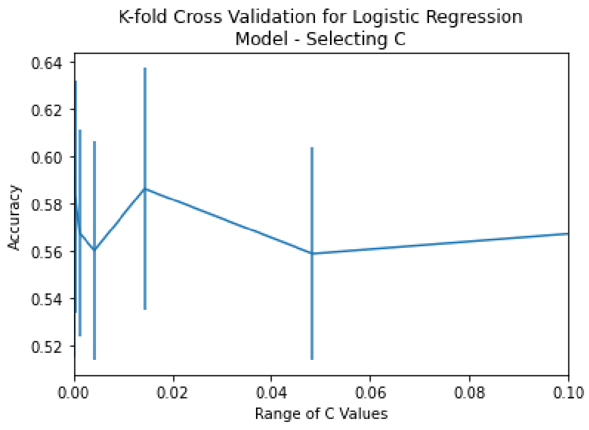

Figure 1d: Errorbar for Logistic Regression (Narrowest C Value Range)

KNN Classifier Implementation

Cross validation was then used again to choose the values of K for the KNN classifier. The range of K ranges chosen for testing were 2 to 10. Similarly to the selection for the logistic regression classifier, the K values were chosen based on mean error. Given the large size of the dataset, the process was computationally slow. The optimal value found ws K=10 which was used for the final model.

Results

In this section, the results from the two approaches discussed above will be examined and evaluated. The methods of evaluation include Accuracy Scores, Classification Reports, Confusion Matrices and ROC Curves. Additionally, the two models will be compared to two baseline models. The baseline classifiers were created using the DummyClassifier method, selecting the most frequent variable and predicting that variable for all inputs. We created two baseline classifiers; the first was trained on the resampled dataset and the second was trained on the original data prior to resampling. Table 3 compares the accuracy and log loss of the baseline classifiers. It is clear that the Baseline Classifier 2 performs far better than the Baseline Classifier 1.

	Baseline Classifier 1	**Baseline Classifier 1**
Testing accuracy	0.11533957845433256	0.8846604215456675
Log Loss	30.555088485193227	3.9836879097174505

Table 3: The testing accuracy and log loss values of the baseline classifiers

Accuracy Scores

The KNN classifier performs better than the logistic regression classifier, with a score of 62.21%. In comparison, the logistic regression classifier only scored 55% accuracy. These are poor performances especially in comparison to the scoring of baseline classifier 2, which had a 88.46% accuracy. An explanation for the poor performance could be related to the resampling. This was done to avoid overfitting the models to the data but it has had the effect of underfitting it too.

Confusion Matrix

Figures 3 and 4 present the confusion matrixes of the respective models. Again like the accuracy scores, the predicted data appears to be overfitted. While there is more true values than false ones, the ratio between the true and false values for logistic regression model is very balanced which is not a desired outcome. The performance for KNN is slightly better with a more severe divide between false and true values. Interesting the training data for KNN appears to have been far more overfitted compared to the logistic regression model.

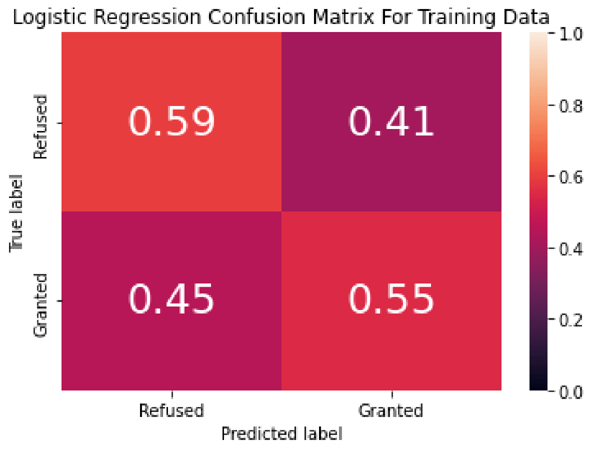

Figure 2a: Logistic Regression Confusion matrix for Training Data

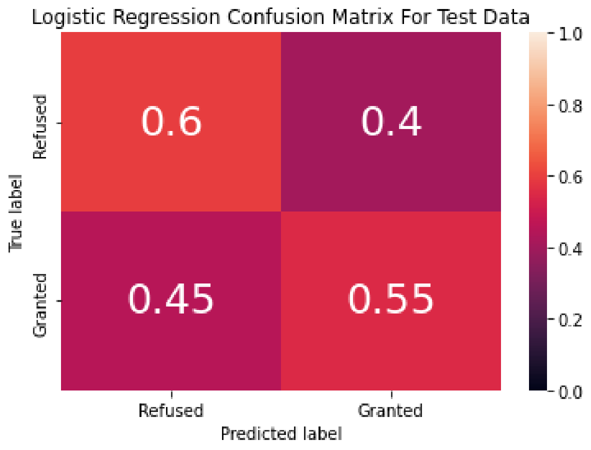

Figure 2b: Logistic Regression Confusion matrix for Test Data

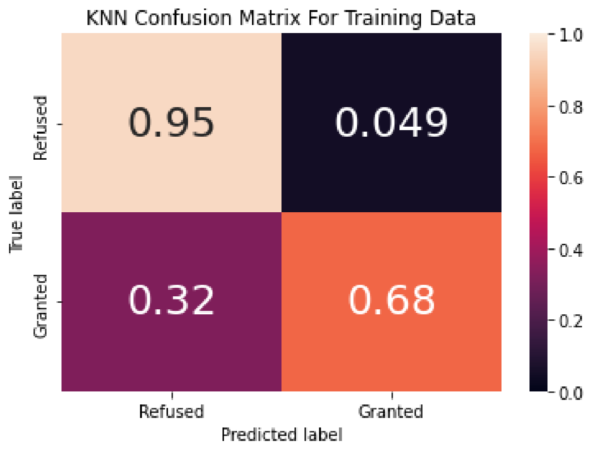

Figure 3a: KNN Confusion matrix for Training Data

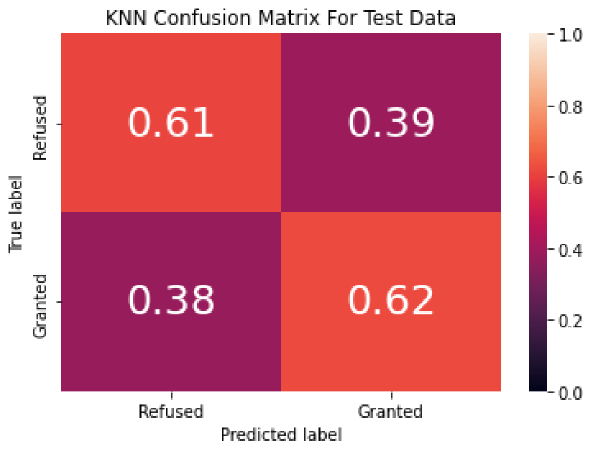

Figure 3b: KNN Confusion matrix for Test Data

Classification reports

A Classification report is a function from the classification_report to measure the quality of predictions from a classification algorithm. The report shows the main classification metrics precision, recall and f1-score on a per-class basis. These metrics are calculated by using true and false positives, true and false negatives. The f1-score combines the precision and recall of a classifier into a single metric by taking their harmonic mean and is therefore a good indication of model performance. For each model, a classification report was produced based on training data and test data. Please know that the training data was resampled data, while the test data was not. Tables 2-5 present the classification reports of the models. Similarly to previous evaluation methods, the models performed poorly here. Unsurprisingly, the KNN classifier again performed better.

	Precision	**Recall**	**f1-score**	**support**
0	0.57	0.59	0.58	48205
1	0.58	0.55	0.56	48205
accuracy			0.57	96410
macro avg	0.57	0.57	0.57	96410
weighted avg	0.57	0.57	0.57	96410

Table 2: Logistic Regression Classification Table for Training Data

	Precision	**Recall**	**f1-score**	**support**
0	0.15	0.60	0.24	1576
1	0.91	0.55	0.69	12088
accuracy			0.56	13664
macro avg	0.53	0.57	0.46	13664
weighted avg	0.82	0.56	0.63	13664

Table 3: Logistic Regression Classification Table for Test Data

	Precision	**Recall**	**f1-score**	**support**
0	0.75	0.95	0.84	48205
1	0.93	0.68	0.78	48205
accuracy			0.81	96410
macro avg	0.84	0.81	0.81	96410
weighted avg	0.84	0.81	0.81	96410

Table 4: KNN Classification Table for Training Data

	Precision	**Recall**	**f1-score**	**support**
0	0.18	0.61	0.27	1576
1	0.93	0.62	0.74	12088

accuracy			0.62	13664
macro avg	0.55	0.62	0.51	13664
weighted avg	0.84	0.62	0.69	13664

Table 5: KNN Classification Table for Test Data

ROC Curve Analysis

The last evaluation method is the ROC Curve. A ROC curve is a plot of the true positive rate against the false positive rate. Fundamentally, the optimal ROC curve is one that comes as close as possible to the top-left corner. Figure 4 is the plot of the ROC Curve for each classifier (logistic regression, KNN classifier) for both the training and test datasets. From observing this plot, the best performance came from the KNN classifier again.

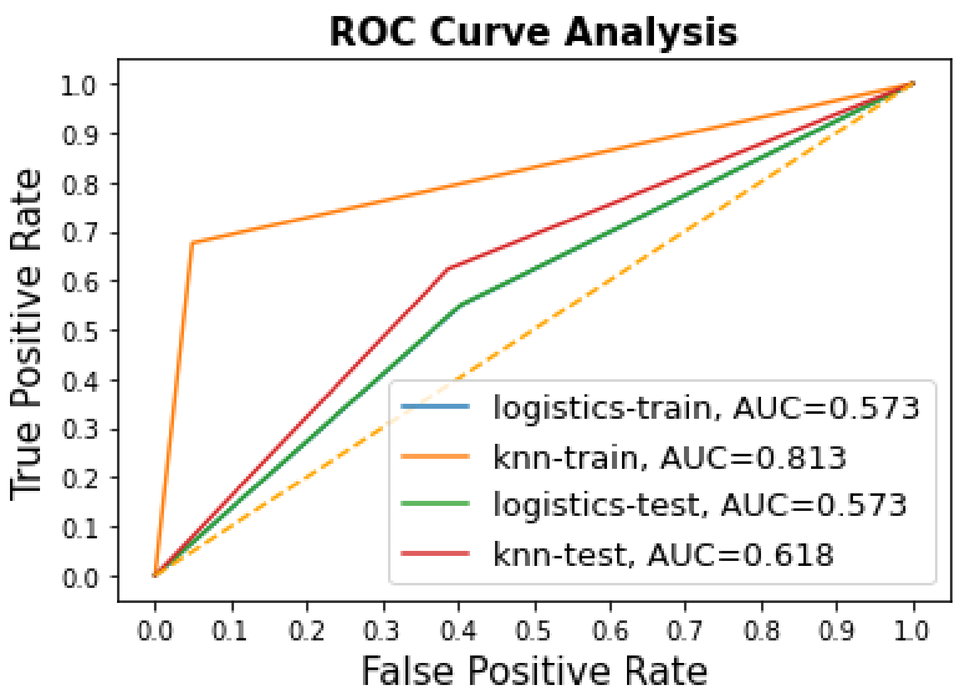

Figure 4: ROC Curve Analysis of Logistic Regression Classifier and KNN Classifier

Given all this evidence, it is safe to conclude that the KNN classifier was the best classifier between it and the logistic regression classifier. However, given the performance of the baseline classifier 2, which is completely overfitted to the training data, there is still more improvement required. There may be a strong case to be made that better features should be selected and an improved method of resampling should be applied to avoid these pitfalls in the future.

Summary

As we have mentioned before, in this report we focus on predicting if a planning permit will be approved or not, for this approach we use classification algorithms such as: Logistic Regression and KNN. We carefully preprocessed the data and used a decision tree for feature importance, once that is done we use SMOTE (Synthetic Minority Over-sampling Technique) to balance the dataset. Additionally, we performed K-Fold Cross Validation to identify the optimal hyperparameter values for both classification algorithms. Once we trained and tested the models we can see that KNN behaves in a better way compared to Logistic Regression.

Contributions

Juan Montenegro :

- Choice of possible datasets for the project
- Creating repository on GitHub
- Write code to model Logistic Regression and KNN
- Write code to feature importance (Decision Tree)
- Write code to model Logistic Regression and KNN
- Write code to Hyperparameter tuning and Cross Validation
- Write code to Training Dummy Classifier
- Write code to metrics and graphs
- Explanation of the methods in the report
- Explanation of Feature Importance in the report

odonneb4@tcd.ie Bríd O'Donnell:

- Choice of possible datasets for the project
- Wrote code to identify the boundary areas of the application sites
- Wrote code for cleaning the data
- Wrote up introduction
- Wrote up explanation of Dataset and Features
- Wrote up experiment/results/discussion section

Github Repo: https://github.com/juanky2797/MLProject