1. Introduction

Our objective is to provide our best-obtained results of various Machine Learning algorithms. Our primary task is to classify Real Estate into different classes based on its price. The dataset used for this task is taken from OpenML's free dataset. To obtain optimal results, we use different kinds of techniques involving the following tasks: handling missing data, feature selection, feature engineering, handling class imbalance, hyperparameter tuning, etc. As a result, Random Forest has shown to provide the highest accuracy compared to other classifiers.

2. Problem Definition and Algorithm

2.1 Task Definition

Our task is to predict the class based on the final price of each home through the usage of given features. We can summarize the task with the given input and output as follows:

Input: 81 different features/indicators that may affect house pricing.

Output: the classes (there are a total of 5 classes).

The classes are further defined as follows:

1. Sale Price between $0 and $100,000
2. Sale Price greater than $100,000 but less than or equal to $200,000
3. Sale Price greater than $200,000 but less than or equal to $300,000
4. Sale Price greater than $300,000 but less than or equal to $400,000
5. Sale Price greater than $400,000

2.2 Algorithm and Techniques Definition

Below are the machine learning techniques we use and their importance:

A. Handling Missing Values

If missing values are not handled properly, it could lead to incorrect results and low precision. There are 2 primary methods to handle missing values which are Imputation and removal of data. Imputation means changing missing values with guesses obtained with mathematical techniques.

B. Data Encoding

Most machine learning algorithms only work with numerical values. Hence we must transform non-numerical values.

C. Feature Selection

It is important that our model is only fed with relevant data. Irrelevant data are considered noise and may reduce model performance.

D. Feature Engineering

With Feature Engineering the raw data is prepared for the specific task at hand. This includes transforming the data, creating new features as well as scaling or normalizing data.

E. Data Sampling

To make correct assumptions about the population without having all the data, the subset that we look at should represent the population, this is done by sampling. On the other hand, if the given dataset is imbalanced, we can use resampling or upsampling to make our dataset more balanced.

F. Hyperparameter Tuning

Hyperparameters in Machine Learning models are features that can be set by hand, before training the model. Having the optimal Hyperparameters increases the effectiveness of the model.

3. Experimental Evaluation

3.1 Methodology

A. Data Pre-processing

a. Handling missing values

In the numerical columns the missing values are replaced by the median. In the categorical columns, the missing values are replaced with 'Unknown' and 'NA'.

b. Data encoding

To encode the categorical columns, the get_dummies function from the pandas library is used.

c. Data sampling - handling class imbalance

Class 1 is found to be overrepresented compared to other classes. Hence, upsampling of data is performed. This is done by using the SMOTE method.

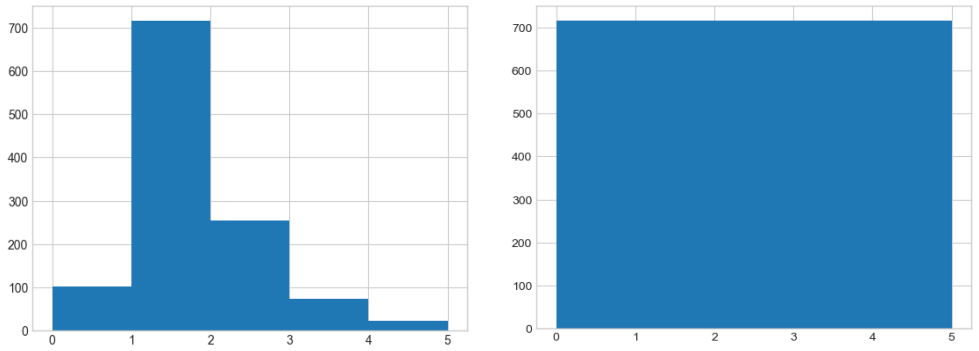

Fig 1. (Left) The frequency of houses in each class before data sampling. (Right) The frequency of houses in each class after data sampling.

d. Feature Engineering

Some features are combined for the creation of new features. In total, we add 8 new features. There are three “Total” features, where other features are added up. Four of the features look at if features exist, like pools or garages. This is done by giving the new variable the value 1 if there is a feature and 0 if there isn’t. The last new feature is concerned with the remodeling and building dates. Correlation checks are then conducted on the newly formed features to check if they are useful. It is shown that the features “TotalSF”, “TotalBath” and “HasFireplace” have the highest correlation with “Class”.

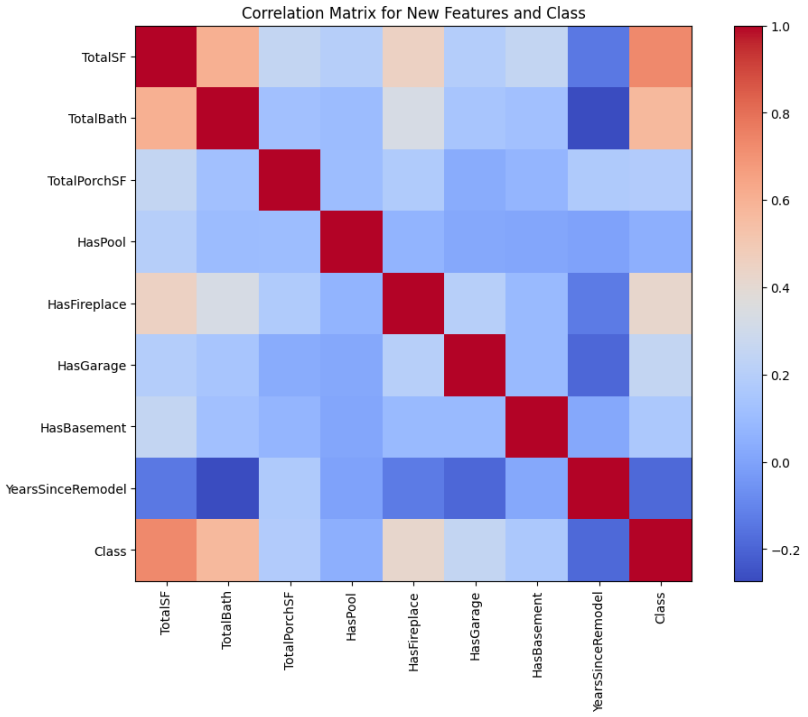

Fig 2. The correlation matrix of new features and class after feature engineering

e. Classifier

The classification is conducted with different kinds of classifiers: KNN, Decision Tree, SVM, Random Forest, etc. Random Forest is the Classifier with the highest accuracy, hence it is used.

f. Hyperparameter Tuning

Best hyperparameters are selected through the GridSearchCV method from Sklearn. The best-found parameters given are as follows:

```
n_estimators=1670,
criterion='gini',
max_depth=18,
```

```
min_samples_split=3,
min_samples_leaf=1,
min_weight_fraction_leaf=0.0,
max_features='sqrt',
max_leaf_nodes=None,
min_impurity_decrease=0.0,
bootstrap=False,
oob_score=False,
n_jobs=-1,
class_weight='balanced',
random_state=42
```

3.2 Results and Analysis

Our model for the classification of houses in Iowa achieves an Accuracy of 0.9041, a Precision of 0.9046, a Recall of 0.9041 and an F1-Score of 0.9014 when looking at the weighted averages. The total confusion matrix looks like this:

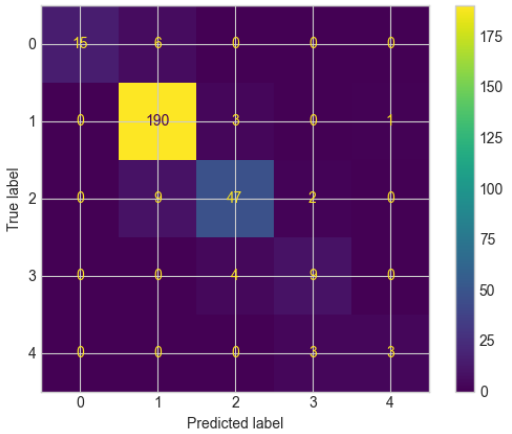

Fig 3. The total confusion matrix for true and predicted labels.

Overall, our results achieve a relatively high accuracy, which means the model is capable of achieving a high number of correct predictions. However, since our data is imbalanced, looking purely at accuracy will be misleading. On imbalanced problems, a high accuracy score is easily achieved by simply classifying all observations as the majority class. Hence, looking at precision and recall is as important. Our achieved recall value is roughly 0.9 which is good, however high recall values often lead to low precision. When the recall value is high, this means that the model has correctly classified values when values are positive but it does not say anything about negative values. This is when precision plays a role. Precision tells them about the possible odds when the model makes a correct prediction. In conclusion, it is important for our model to balance both precision and recall. Our model has received a relatively balanced score between accuracy, recall, and precision which made us convinced that our model will work well.

However, we do also see a small problem. If we see the confusion matrix, for class 4 we can see the number of correct predictions and wrong predictions is almost the same. This may be due to the small amount of data provided. Hence, further analysis is needed.

3.3 Discussion

Missing data is the first thing we handled. This is because we are not able to train models with NaN values. Hence, it needs to be handled first. Methods of handling could be different. We realized that different types of data (categorical, numerical, etc.) should be handled differently for optimal results. We also tried classifying some numerical features as categorical, such as ['MSSubClass', 'OverallQual', 'OverallCond'], but this led to a worse performance.

Next, various trials among different classifiers were done which include SVC, Decision Trees, KNN, and Random Forest. Among them, however, SVC and Random Forest have shown to perform better compared to the rest. However, after doing a few hyperparameter tuning with Grid Search, Random Forest showed better accuracy. Hyperparameter tuning, in general, did not increase the accuracy by much.

Interestingly, we found that feature engineering has shown to have a positive effect. It improves the correlation of features and target class which has an effect on our final results. Below shows the metric difference with and without feature engineering:

	Precision	Recall	F1-Score
without Feature Engineering	0.8874	0.8870	0.8835
with Feature Engineering	0.9046	0.9041	0.9014

Table 1. Metrics differences when Feature Engineering is used

Lastly, it is important to take good care of class imbalance. Even though accuracy might seem high, there is a high chance of overfitting with lower precision and recall values.

	Precision	Recall	F1-Score
without Data Upsampling	0.8796	0.8801	0.8747
with Data Upsampling	0.9046	0.9041	0.9014

Table 2. Metrics differences when Data Upsampling is used

6. Conclusion

Data pre-processing and feature engineering are the two main important factors of our algorithm. There was a significant difference with and without the two techniques. Lastly, tackling class imbalance by upsampling the minority classes is a good technique to reduce the risk of overfitting.