Abstract

The paper describes how to identify fraudulent vehicle insurance claim applications using machine learning approaches. The Scikit-Learn package was used to train a number of classification models using the Vehicle Insurance Claim Fraud Detection dataset. The models' performance was enhanced using a variety of methods, such as feature engineering and hyperparameter tuning. The results demonstrate that these algorithms can identify false claims, which can lead to significant cost savings for insurance companies.

Introduction

Vehicle insurance fraud is a common problem that can cost insurance companies a lot of money. Machine learning algorithms can be used to precisely identify fraudulent claims in order to reduce these losses. The Vehicle Insurance Claim Fraud Detection dataset, which includes details on roughly 32 features relevant to each claim submission, is used in this study. These aspects include details about the claimant, the insurance, the car, and the accident.

I use the Scikit-Learn package to create a number of classification models to determine whether a claim submission is fraudulent or not. For example, we investigate different classification techniques including Logistic Regression, Support Vector Machines (SVM), Decision Trees, Random Forest, K-Nearest Neighbours (KNN), Naive Bayes, and LightGBM. Additionally, we use a variety of methods, such as feature engineering and hyperparameter tuning, to improve the performance of our models.

Utilising stratified cross-validation, which guarantees that each fold of the data contains a representative sample of both fraudulent and non-fraudulent claims, we assess the performance of our models. We also talk about the problem of class imbalance in the dataset, which can affect how well our models work. We investigate various methods, such as oversampling and undersampling strategies, to address this problem.

The objective of this project is to provide insurance companies a more reliable and effective way for identifying fraudulent claims. We seek to provide a complete approach for identifying bogus claims and ultimately reduce the financial losses suffered by insurance firms by applying a variety of classification algorithms and optimising their performance through various techniques.

1 EDA

This data set has 33 columns and 15,420 entries. The target variable is "FraudFound-P" in the column. Its values show if the claim was fabricated or not. 5.99% of the claims are fraudulent, compared to 94.01% of the total claims.

For the purpose of training a model, the following categorical variables must be transformed into numerical variables: Month, Day of Week, Make, Accident Area, Day of Week Claimed, Month Claimed, Sex, Marital Status, Fault, Policy Type, Vehicle Category, Vehicle Price, Days Policy Accident, Days Policy Claim, Past Number of Claims, Age of Vehicle, Age of Policy Holder, Police Report Filed, Witness Present, Agent Type, Number of Suppliments, Address Change-Claim, Number of

Some columns, such as "Days-Policy Accident," "Days Policy Claim," "Past Number of Claims," "Age Of Vehicle," "Age Of Policy Holder," "Number Of Suppliments," "Number Of Cars," and "Vehicle Price," appear to have the incorrect data types assigned to them. They should all be numbered.

The analysis you supplied provides a bit of insight on how claims are distributed by several categories, including month, make, accident region, sex, and marital status, as well as how fraud claims are distributed within each group.

Key findings include:

- The most claims are filed in January, and the least in August.
- Pontiac, Toyota, and Honda are the three automobile brands that appear the most frequently in the dataset, while Ferrari, Lexus, and Mercedes are the three that appear the least frequently.
- Urban areas are where the majority of claims (92.01%) are from.
- The majority of drivers involved in accidents (73.21%) are men, and 6.29% of men who filed claims were false.
- Compared to other marital status groupings, married people had the greatest claims.
- By month, no patterns of fraud claims were found.
- By make, no patterns of fraud claims were found.
- Females are more likely than males to file fraudulent claims.
- There were no similarities in the fraud claims broken down by marital status.

Overall, these findings may help insurance firms to better target their fraud detection efforts at particular types of claims. For instance, if ladies submit fraud claims more frequently than males, insurance companies might look into female claims more extensively or concentrate their anti-fraud awareness campaigns on women.

2 Data pre-processing and encoding

Encoding categorical features into numerical values is a typical data preprocessing technique used in machine learning, as seen in this section's code. The algorithm locates all object-type features in a dataset and converts them into numerical values using several mapping strategies, such as substituting numerical values for categorical texts. The accuracy of machine learning algorithms can be increased by transforming category data into

numerical values, which is why this technique is crucial as many of them require numerical inputs.

The challenge of fraud detection is the main emphasis of data pre-processing. The data set's categorical characteristics must first be located and added to a list named object-type-feature in the first step. The replace() function is then used to translate each categorical value to a matching numerical value, encoding the categorical features into numerical values. Using the train-test-split() tool from Scikit-Learn, the data set is finally divided into training and test sets, with the training set containing 80% of the data and the test set containing the remaining 20%. In order to train and test the model, the generated data sets are placed in the variables X-train, X-test, Y-train, and Y-test.

3 Model Development

3.1 Logestic Regression

For binary classification issues, a common approach is logistic regression. It is a kind of generalised linear model that calculates the likelihood of an event happening from a collection of input data by fitting a logistic curve to the data. A sigmoid function is used in logistic regression to convert the result into a probability value between 0 and 1, presuming that the connection between the input factors and the output variable is linear. In this manner, logistic regression can be used to forecast the likelihood of a binary result, such a customer's purchase of a product.

The logistic regression model finds the best-fit line that divides the two classes of data by estimating the coefficients of the input variables. These coefficients are estimated by the model using the maximum likelihood estimation approach, and predictions are then made using new data. Any predictions with a probability greater than 0.5 are classified as belonging to one class, and those with a probability less than 0.5 are classified as belonging to the other class. Typically, the decision boundary separating the two classes is set at a probability threshold of 0.5. In general, logistic regression is a straightforward, easily understood, and effective technique that may be used to a number of binary classification issues.

3.2 SVM

A popular supervised learning technique for classification and regression applications is called Support Vector Machines (SVM). Finding the optimum hyperplane that divides the data into various classes is the goal of SVM. In order to move the data into a higher dimensional space where it is easier to determine the optimum hyperplane to divide the data, SVM employs a variety of kernels.

The linear kernel, which is merely the dot product of the input characteristics, is one of the most often used kernels. When the data can be divided into two classes by a straight line or a hyperplane, or when the data can be linearly split, the linear kernel performs well. The linear kernel might not be appropriate, though, if the data are not linearly separable.

Another common kernel in SVM is the polynomial kernel. A polynomial function is used by the polynomial kernel to translate the input data into a higher dimensional space. The degree of the transformation depends on the degree of the polynomial. When the data contains non-linear properties, the polynomial kernel is advantageous.

The input data are mapped into a sigmoid function by the non-linear sigmoid kernel. When the data contains non-linear features, the sigmoid kernel can be employed for classification tasks.

The radial basis function (RBF) kernel is yet another well-liked SVM kernel. A Gaussian function is used by the RBF kernel to turn the input data into an infinitely dimensional space. When the data cannot be separated linearly and includes intricate non-linear properties, the RBF kernel is helpful. Due to its adaptability and efficiency, it is frequently used as the default kernel in SVM.

3.3 Decision Trees

A popular machine learning approach called decision trees can be applied to classification and regression tasks. The programme creates a model of decisions and potential outcomes that resembles a tree. Each leaf node of the tree represents a class label or a numeric value, whereas each interior node indicates a decision based on a particular characteristic or attribute. The objective is to build a tree with high accuracy and good generalizability to new data.

Decision trees have the benefit of being simple to understand and analyse, which makes them ideal for outlining the thought process behind a prediction. They are tolerant of outliers and missing values and can handle both category and numerical data. They can, however, overfit, which leads to the creation of excessively complex trees that perform well on training data but poorly generalise to fresh data. Pruning, establishing the minimum amount of samples needed to divide an internal node, and establishing the maximum depth for the tree are all methods that can be utilised to avoid overfitting.

There are various Decision Tree algorithms, including ID3, C4.5, and CART, each with unique features and benefits. Decision Tree performance can also be enhanced by ensemble techniques like Random Forest and Gradient Boosted Trees.

3.4 Random Forest

Popular machine learning algorithm Random Forest is used for classification and regression tasks. It is an ensemble learning technique that builds a number of decision trees from a series of randomly chosen data subsets, combining the results to get a final prediction. The method creates a number of decision trees, each of which is made up of a randomly selected part of the training data and a randomly selected subset of the features.

To lessen the variance of the model, each decision tree is grown to its maximum depth during training without being pruned. Each tree in the forest independently classifies the input while generating a forecast, and the classification that receives the most votes determines the outcome.

Random Forest is well suited for high-dimensional datasets due to its ability to allow a large number of input features without overfitting, which is one of its main advantages. Furthermore, Random Forest has the ability to offer feature importance scores, which can be helpful for determining the model's most crucial variables. The large number of decision trees that must be built, however, makes Random Forest potentially computationally expensive to train on large datasets.

3.5 KNN

A simple yet effective machine learning approach called K-Nearest Neighbours (KNN) is used for classification and regression problems. In KNN, a new instance's classification is determined by the classes of its k nearest neighbours in the feature space. K's value is a hyperparameter that must be adjusted based on the given data and problem. KNN is a non-parametric algorithm, which means it doesn't assume anything about how the data

are distributed. Because it doesn't create a model during training but instead stores all the training data in memory and uses it during testing, it is also a lazy algorithm. It is often important to normalise the data before applying KNN because the method can be sensitive to the magnitude of the features. Since KNN determines the distance between every pair of instances in the dataset, it can be computationally expensive for large datasets.

One of KNN's benefits is that it can function well even when the data is noisy or the decision boundary is extremely unpredictable. It can be used for both classification and regression tasks and can handle multi-class classification problems. The performance of the technique degrades as the number of features rises, and this is known as the "curse of dimensionality" for KNN. Furthermore, picking an appropriate value for k can be difficult, and doing so can affect how well the algorithm performs.

3.6 Naive Bayes

A popular probabilistic approach for classification issues is naive Bayes. It is based on the Bayes theorem, according to which the likelihood of a hypothesis H (in this case, a class label) given some observed evidence E (the characteristics of an instance) is proportional to the likelihood of E given H, multiplied by the likelihood of H in the past. The term "naive" refers to the assumption that, given the class label, the features are conditionally independent, meaning that the presence or absence of one feature has no bearing on the likelihood of the other traits. Naive Bayes is often used as a basis model in text categorization and spam filtering tasks, where the assumption of independence may hold relatively well. This is a simple assumption that can result in less accurate predictions.

There are three primary types of Naive Bayes classifiers: Gaussian Naive Bayes, Multinomial Naive Bayes, and Bernoulli Naive Bayes. Naive Bayes can handle both continuous and categorical information. While Multinomial Naive Bayes and Bernoulli Naive Bayes are intended for discrete features, such as word counts in text classification, Gaussian Naive Bayes assumes that continuous features are normally distributed. While Bernoulli Naive Bayes is suitable for binary classification problems, where each feature can only take on two values (for example, presence or absence of a word), Multinomial Naive Bayes is frequently used for document classification tasks.

3.7 LGBM

An effective and scalable gradient boosting framework is called LightGBM (LGBM). It uses histogram-based algorithms rather than more conventional bin-based algorithms to shorten training times and conserve memory. Tree-based learning techniques are used to segment data by features. Gradient-based One-Side Sampling (GOSS), another method used by LGBM, minimises the amount of samples required for gradient-based learning, cutting down on training time.

In comparison to conventional gradient boosting techniques, LGBM has a number of advantages, such as improved support for large-scale datasets, improved accuracy, and quicker training periods. The number of leaves per tree, the pace of learning, and the maximum depth of the tree are just a few of the numerous hyperparameters that can be adjusted to optimise performance. Due to its outstanding performance and scalability, LGBM has emerged as a prominent method in numerous machine learning contests and real-world applications.

4 Model Evaluation

We evaluate the effectiveness of various machine learning techniques that we applied to our issue. In this scenario, we applied Naive Bayes, LightGBM, K-Nearest Neighbours (KNN), Random Forest, Decision Trees, Support Vector Machine (SVM), and Logistic Regression.

In order to compare the performance of various algorithms, we employed evaluation criteria including accuracy, precision, recall, F1-score, and ROC-AUC. The most effective algorithm for a given problem relies on the type of data being used and the particular task at hand. Each method has advantages and disadvantages.

In our instance, we tested the models on a dataset and chose the algorithm with the greatest performance based on the assessment measures. The evaluation's findings are displayed in a table so that we can compare each algorithm's performance and choose the most effective solution for our issue.

Method	training acuracy	test accuacy	model accuracy	MAE
Logestic Regression	0.9396523179	0.9440397351	0.9405298013	0.05947019868
SVM(linear)	0.9368377483	0.9394039735	0.9373509934	
SVM(poly)	0.9399834437	0.9447019868	0.9409271523	
SVM(sigmoid)	0.8947847682	0.8950331126	0.8948344371	
SVM(RBF)	1	0.9447019868	0.9889403974	
Decision Trees	1	0.9175496689	0.9835099338	0.01649006623
Random Foret	0.9408112583	0.9447019868	0.941589404	0.05841059603
Random Forest(M)	0.9999172185	0.9450331126	0.9889403974	0.01105960265
KNN	0.9427152318	0.938410596	0.9418543046	0.05814569536
Naive Bayes	0.9026490066	0.8993377483	0.901986755	0.09801324503
LGBM	0.9428807947	0.9473509934	0.9437748344	0.05622516556

Figure 1: Results

5 Model Optimization

Model optimisation involves modifying a model's hyperparameters to enhance performance. You tuned the hyperparameters of the SVM and Random Forest models in this project to maximise their performance. Hyperparameters are movable parameters that are set by the user rather than being learned during training.

The kernel type was tuned as a hyperparameter for SVM. The type of decision boundary that the SVM utilises to divide the classes is specified by the kernel function. The balance between increasing the margin and reducing the classification error is controlled by the regularisation parameter. You were able to determine the ideal configuration that yielded the greatest results by experimenting with various combinations of these hyperparameters.

The number of trees, the maximum depth of each tree, and the minimal amount of samples needed to divide a node are some of the hyperparameters for Random Forest that have been modified. You were able to design an ensemble of decision trees with a higher prediction accuracy by modifying these hyperparameters.

Model optimization is a crucial stage in machine learning that can significantly enhance a model's performance. The trade-off between model complexity and performance must be

balanced, though, as highly complicated models may suffer from overfitting and be unable to generalise to new data.

You can see the changed results of SVM and Random Forest as SVM(RBF) and Random Forest(M) in the previous table.

6 stratified cross-validation

K-fold stratification By dividing the data into k-folds, each fold is a subset of the data that is used as the testing set while the remaining folds are used as the training set, a technique known as cross-validation is used to assess the performance of a machine learning model. When working with unbalanced datasets, the stratified component of the cross-validation technique ensures that the proportion of target classes is the same in each fold.

The data is divided into 5 stratified folds in the code I offered.A logistic regression model is then trained on the training set and tested on the testing set for each fold. The mean and standard deviation of the accuracy ratings for each fold are computed at the end and saved in cv-scores.

Any machine learning algorithm can use this cross-validation technique, which is especially helpful for evaluating the performance of various models and choosing the ideal hyperparameters for a model. We may obtain a more accurate assessment of how well a model is expected to perform on brand-new, untested data by using many folds and computing the mean and standard deviation of the accuracy scores.

Method	Mean accuracy	Standard deviation
SVM RBF	0.9447019868	0
Decision-Trees	0.8562251656	0.0270344721
Random Forest	0.9447019868	0
KNN	0.9400662252	0.005662154834
Naive Bayes	0.9371523179	0.00788997184
LGBM	0.9447019868	0

Figure 2: Stratified Results

7 Imbalanced data challenge

In real-world situations, where the proportion of one class is noticeably higher than the other, unbalanced datasets are quite prevalent. In such cases, the model can start to favour the majority class and do poorly when applied to the minority class. The dataset is unbalanced in this instance because there are much fewer fraud cases than non-fraud ones.

Using sample strategies like under- or over-sampling is one approach to address this problem. In general:

1. Increasing the number of instances in the minority class is a method known as oversampling. This can be accomplished in a number of ways, either by creating synthetic instances or by copying already-existing instances. Synthetic Minority Over-sampling Technique (SMOTE) is a popular oversampling technique. SMOTE functions by producing artificial samples that are similar to but not identical to minority class samples that already exist.

2. On the other hand, undersampling entails lowering the proportion of occurrences in the majority class. This can be accomplished by choosing randomly from a subset of instances belonging to the majority class or by choosing examples that are comparable to minority class instances. RandomUnderSampler is a popular undersampling technique that randomly selects a portion of majority class samples to keep in the dataset.

There are benefits and drawbacks to both oversampling and undersampling methods. Undersampling can lead to information loss, while oversampling can increase the danger of overfitting. Consequently, it's frequently advised to try both techniques and evaluate their effectiveness in order to find the best answer for the particular problem and dataset.
Additionally, it's crucial to keep in mind that there are other approaches for handling unbalanced datasets, such as cost-sensitive learning, which entails giving the minority class more weight during training. Another way for improving classifier performance on unbalanced datasets is to use ensemble algorithms, such as bagging or boosting.
Resampling methods are used to tackle unbalanced datasets, and they include RandomOverSampler and RandomUnderSampler. While RandomUnderSampler reduces the samples of the majority class by randomly removing them, RandomOverSampler increases the samples of the minority class by randomly replicating them.

I trained the Logistic Regression and Random Forest models on the resampled training data after balancing the classes, and I then assessed how well they performed on the test set. Below are the outcomes.

Method	training accuracy	test set accuracy	model accuracy
Logestic Regression	0.7282758621	0.6847682119	0.6822516556
Random Forest	0.8931034483	0.4258278146	0.5014569536

Figure 3: Under Sampling

Method	training accuracy	test set accuracy	model accuracy
Logestic Regression	0.7282758621	0.6847682119	0.6822516556
Random Forest	0.8584764421	0.6612582781	0.7256291391

Figure 4: Over Sampling

8 Conclusion

I used a number of classification models in this project, including Logistic Regression, SVM, Decision Trees, Random Forest, KNN, Naive Bayes, and LGBM, to identify fraudulent vehicle insurance claims. I employed a number of methods, including stratified cross-validation, over-sampling, and under-sampling, to address the problem of class imbalance. After evaluating each model, I discovered that SVM and Random Forest performed the best. I tuned the hyperparameters to further improve their performance. The findings show that the model is capable of accurately identifying bogus car insurance claims, giving insurance firms a useful tool to increase their fraud detection and prevention capacities.