# Analysis of Machine Learning Algorithms on Fraud Prediction Data

Team: Raghav Jindal,Shivam Bhardwaj,Tejaswi Dabas

# Introduction

Warranty claims are an important aspect of the after-sales service provided by businesses. Predicting the probability of a customer filing a warranty claim can help businesses plan their inventory and allocate resources accordingly. In this project, we will use various machine learning (ML) algorithms to predict the probability of a customer filing a warranty claim for an item sold.

# Objective

Machine learning (ML) is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention. In this project, we will study basic machine learning background, techniques, and related language, and implement either Gaussian Naive Bayes, Decision Tree Classifier, K-Neighbors Classification,Logistic Regression, Random Forest Classifier and Linear Support Vector Classification

# About the Dataset

This page is having the data related to an organization with 20 variables. In that 19 are independent variables to predict the dependent variable 'fraud' with 2 categories '1' indicates fraudulent claim and '0' indicates genuine claim. This data is having 11917 observations.

Using this data set details we have to predict whether a claim is a Genuine claim or fraudulent claim.

# Data-Fields

- Region - Customer region details
- state - Current location of customer
- Area - Area_Urban/rural
- City- Customers current located city
- Consumer_profile- Customer's work profile
- Product_category- Product category
- Product_type- Type of the product_Tv/Ac
- AC_1001_Issue- 1001 is failure of Compressor in AC
- AC_1002_Issue- 1002 is failure of Condenser Coil in AC
- AC_1003_Issue- 1003 is failure of Evaporator Coil in AC
- TV_2001_Issue- 2001 is failure of power supply in Tv
- TV_2002_Issue- 2002 is failure of Inverter in Tv
- TV_2003_Issue- 2003 is failure of Motherboard in Tv
- claim_value- Customer's claim amount in Rs
- Service_Centre- 7 Different service centers
- Product_Age- Duration of the product purchased by customer
- Purchased_from- From where product is purchased
- Call_details- call duration in mins
- Purpose- Purpose_compliant-Compliant raised by customer claim- claimed for the product Other- Other categories out of this
- Fraud- '1'- fraudulent claim, '0' Genuine claim

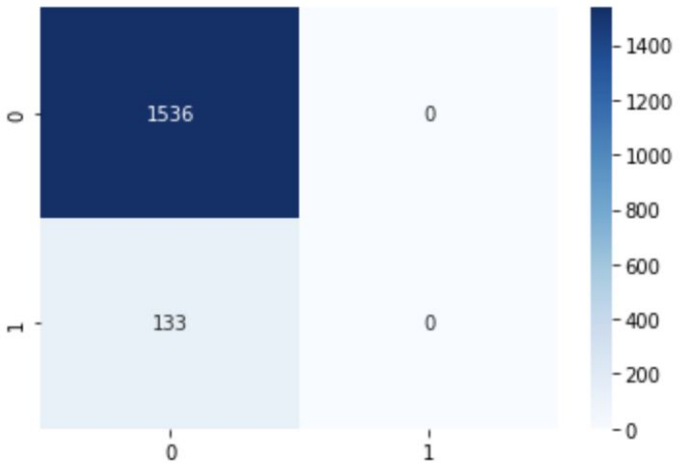
# Logistic Regression

Logistic regression is a statistical method used to model the probability of a binary outcome based on one or more predictor variables. It is commonly used in machine learning and data analysis for classification tasks. The model estimates the parameters of a logistic function that maps the predictor variables to the probability of the outcome, and can be regularized to prevent overfitting. Logistic regression has the advantages of simplicity, interpretability, and the ability to handle both categorical and continuous predictor variables, but may not be suitable for modeling nonlinear relationships.

- Train Time Complexity=O(n*m)
- Test Time Complexity=O(m)
- Space Complexity = O(m)

# Logistic Regression

```
[ ]  cf(log)
```



# Decision Tree Classifier

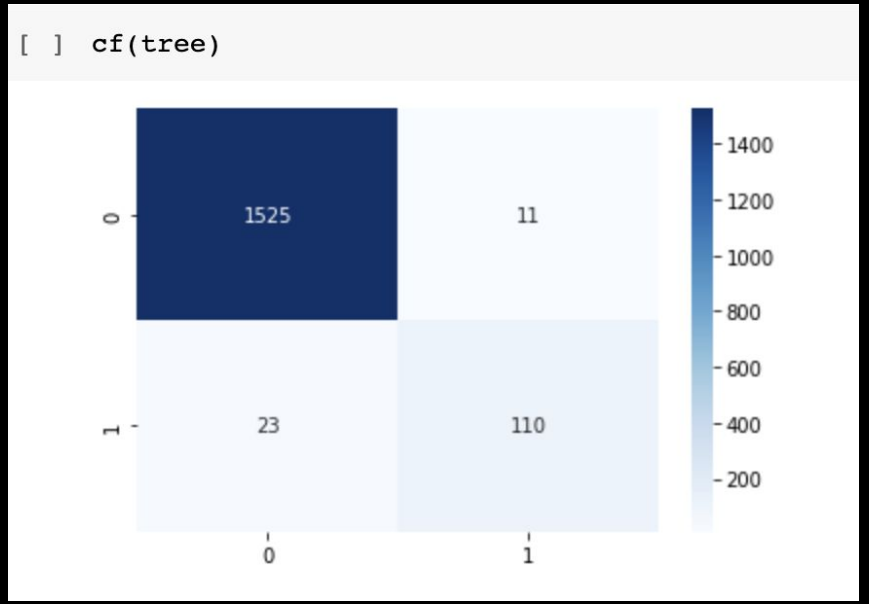
A decision tree classifier is a type of supervised machine learning algorithm used for classification tasks. It works by recursively partitioning the feature space into regions based on the values of the predictor variables, and assigning a label to each region based on the majority class of the training examples that fall within it. The resulting tree-like structure can be used to make predictions for new examples by traversing the tree from the root node to a leaf node corresponding to the predicted class. Decision trees can handle both categorical and continuous predictor variables, and can also capture nonlinear relationships.

Train Time Complexity=O(n*log(n)*m)

- Test Time Complexity=O(m)
- Space Complexity = O(depth of tree)

# Decision Tree Classifier

```
[ ]  cf(tree)
```

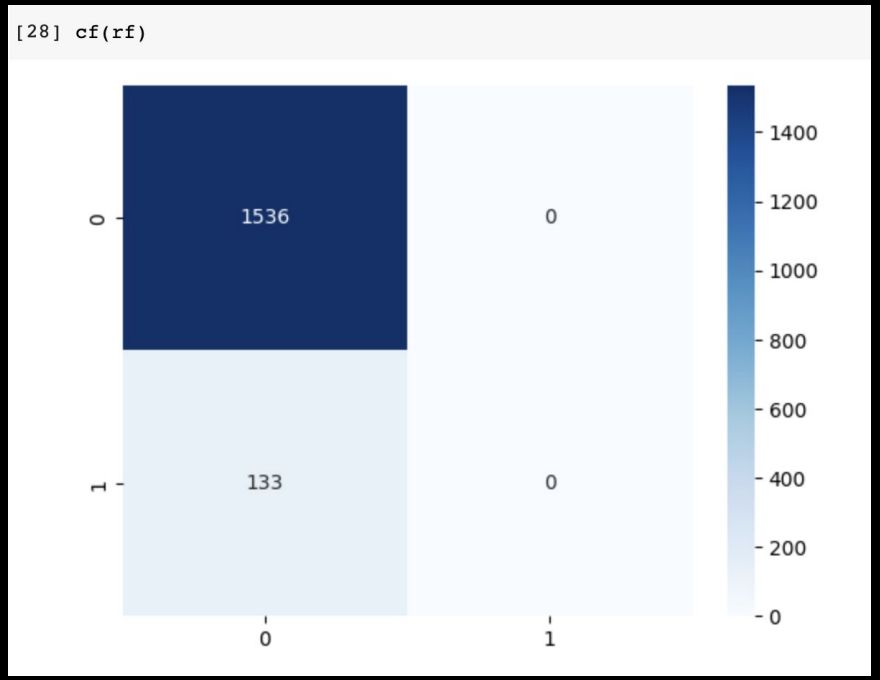


# Random Forest Classifier

A random forest classifier is a type of ensemble learning algorithm used for classification tasks. It works by constructing multiple decision trees on randomly sampled subsets of the training data and features, and aggregating their predictions to make a final prediction. This can improve the accuracy and reduce overfitting compared to a single decision tree. The random forest algorithm can handle both categorical and continuous predictor variables, and can capture complex nonlinear relationships. However, it may require more computational resources than a single decision tree, and may not perform well on imbalanced datasets.

- Train Time Complexity=O(k'*n*log(n)*m)
- Test Time Complexity=O(m*k')
- Space Complexity = O(k'*depth of tree)

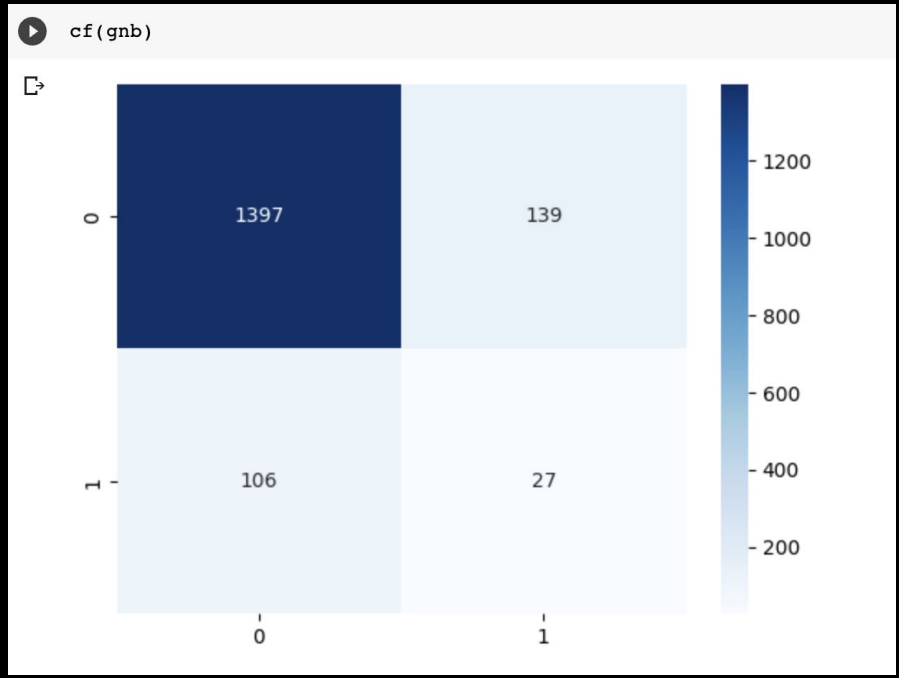
# Random Forest Classifier



# Gaussian Naive Bayes

- Gaussian Naive Bayes is a probabilistic machine learning algorithm used for classification tasks. It is based on Bayes' theorem and the assumption that the predictor variables are independent given the class label. The algorithm estimates the parameters of a Gaussian distribution for each predictor variable and each class, and computes the posterior probability of each class given the predictor values using Bayes' theorem. The predicted class is the one with the highest posterior probability. Gaussian Naive Bayes can handle both categorical and continuous predictor variables, and is computationally efficient and scalable.
- Training Time Complexity = O(n*m)
- Test Time Complexity=O(m)
- Run-time Complexity = O(c*m)

# Gaussian Naive Bayes

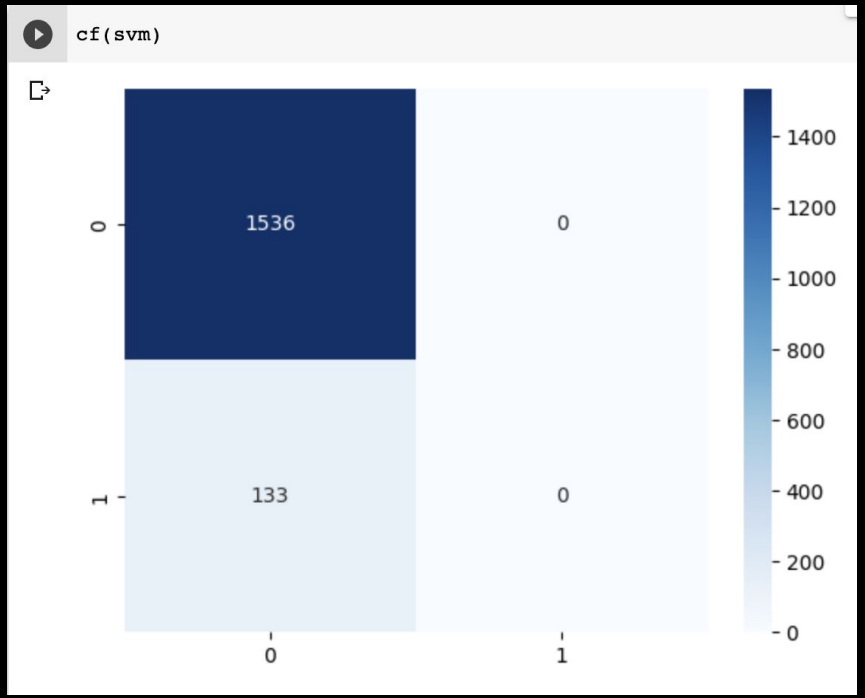


# Linear Support Vector Classification

Linear Support Vector Classification (SVC) is a supervised machine learning algorithm used for classification tasks. It works by finding a hyperplane that separates the data into two classes with the maximum margin, which is the distance between the hyperplane and the closest data points of each class. The hyperplane is defined by a linear combination of the predictor variables, and the algorithm can handle both binary and multiclass classification problems. The linear SVC algorithm can handle both categorical and continuous predictor variables, and is particularly effective when the data is linearly separable or when there is a large margin between the classes.

- Train Time Complexity=O(n^2)
- Test Time Complexity=O(n'*m)
- Space Complexity = O(n*m)

# Linear Support Vector Classification

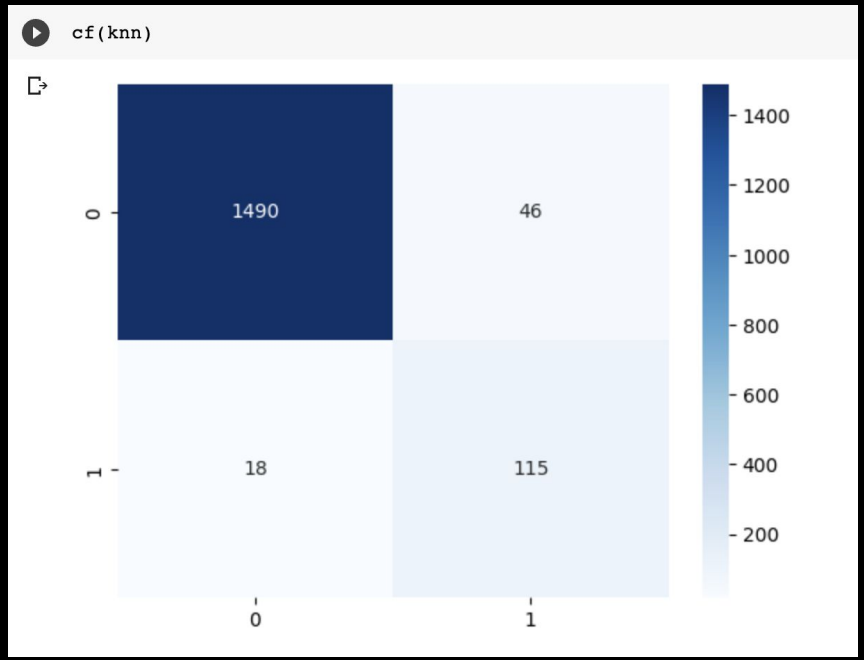


# K-Neighbors Classifier

The K-Nearest Neighbors (KNN) Classifier is a machine learning algorithm used for classification tasks. It works by finding the K closest training examples in the feature space to a new example, and assigning the new example to the majority class of its K nearest neighbors. The KNN algorithm can handle both categorical and continuous predictor variables, and is a non-parametric algorithm that does not make assumptions about the underlying distribution of the data. The performance of the KNN algorithm can be affected by the choice of K, the distance metric used to measure similarity between examples, and the weighting scheme used to assign importance to the neighbors.

- Train Time Complexity=O(k*n*m)
- Test Time Complexity=O(n*m)
- Space Complexity = O(n*m)

# K-Neighbors Classifier



# Result

| | True Positive | False Positive | False Negative | True Negative | True Positive Rate | False Positive Rate | Accuracy |
|---|---|---|---|---|---|---|---|
| **Gausian Naive Bayes** | 1397 | 106 | 27 | 139 | 0.981039 | 0.432653 | 0.837028 |
| **Decision Tree** | 1525 | 23 | 110 | 11 | 0.932722 | 0.676471 | 0.913721 |
| **K-NN** | 1490 | 18 | 115 | 46 | 0.928349 | 0.281250 | 0.892750 |
| **Linear SVM** | 1536 | 133 | 0 | 0 | 1.000000 | 1.000000 | 0.920312 |
| **Logisitic Regression** | 1536 | 133 | 0 | 0 | 1.000000 | 1.000000 | 0.920312 |
| **Random Forest Classifier** | 1536 | 133 | 0 | 0 | 1.000000 | 1.000000 | 0.920312 |

# Result

```python
for model in models:
    scores = cross_val_score(model, X_test, y_test, scoring='accuracy', cv=cv)
    print('Min Accuracy of {}: %.3f'.format(str(model)) % (scores.min()))
    print('Max Accuracy of {}: %.3f'.format(str(model)) % (scores.max()))
    print('Mean Accuracy of {}: %.3f'.format(str(model)) % (np.mean(scores)))
```

```
Min Accuracy of GaussianNB(): 0.778
Max Accuracy of GaussianNB(): 0.867
Mean Accuracy of GaussianNB(): 0.811
Min Accuracy of DecisionTreeClassifier(random_state=39): 0.964
Max Accuracy of DecisionTreeClassifier(random_state=39): 0.994
Mean Accuracy of DecisionTreeClassifier(random_state=39): 0.975
Min Accuracy of KNeighborsClassifier(n_neighbors=3): 0.940
Max Accuracy of KNeighborsClassifier(n_neighbors=3): 0.988
Mean Accuracy of KNeighborsClassifier(n_neighbors=3): 0.961
Min Accuracy of LogisticRegression(max_iter=10000, random_state=39): 0.916
Max Accuracy of LogisticRegression(max_iter=10000, random_state=39): 0.922
Mean Accuracy of LogisticRegression(max_iter=10000, random_state=39): 0.920
Min Accuracy of RandomForestClassifier(max_depth=2, random_state=39): 0.916
Max Accuracy of RandomForestClassifier(max_depth=2, random_state=39): 0.922
Mean Accuracy of RandomForestClassifier(max_depth=2, random_state=39): 0.920
Min Accuracy of LinearSVC(dual=False, random_state=39): 0.916
Max Accuracy of LinearSVC(dual=False, random_state=39): 0.922
Mean Accuracy of LinearSVC(dual=False, random_state=39): 0.920
```

# Conclusion

In conclusion, the Decision Tree Classifier is the best algorithm for predicting genuine warranty claims, based on its high accuracy and low time and space complexity.