

[Home](#) > [Blog](#) > [Machine Learning](#)

Classification in Machine Learning: An Introduction

Learn about classification in machine learning, looking at what it is, how it's used, and some examples of classification algorithms.

Sep 2022 · 14 min read



Zoumana Keita

Senior Data Scientist at International Finance Corporation; Co-founder at ETP4Africa

TOPICS

Machine Learning

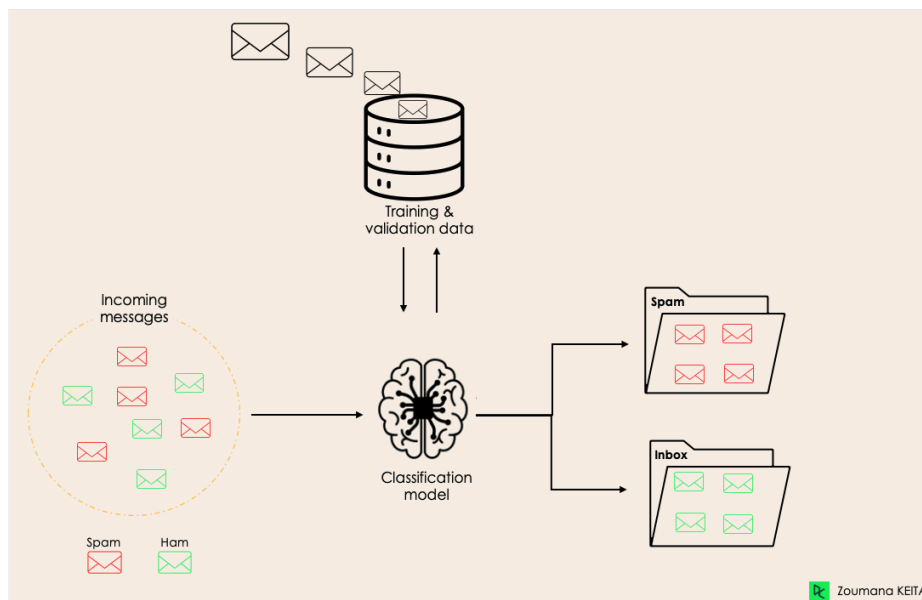
Nowadays, many industries have been dealing with very large data sets of different types. Manually processing all that information can be time-consuming and might not even add value in the long term. Many strategies, from simple automation to machine learning techniques, are being applied for a better return on investment. This conceptual blog will cover one of the most important concepts; classification in machine learning.

We will start by defining what classification is in [Machine Learning](#) before clarifying the two types of learners in machine learning and the difference between classification and regression. Then, we will cover some real-world scenarios where classification can be used. After that, we will introduce all the different types of classification and deep dive into some examples of classification algorithms. Finally, we will provide hands-on practice on the implementation of a few algorithms.

What is Classification in Machine Learning?

Classification is a supervised machine learning method where the model tries to predict the correct label of a given input data. In classification, the model is fully trained using the training data, and then it is evaluated on test data before being used to perform prediction on new unseen data.

For instance, an algorithm can learn to predict whether a given email is spam or ham (no spam), as illustrated below.



Before diving into the classification concept, we will first understand the difference between the two types of learners in classification: lazy and eager learners. Then we will clarify the misconception between classification and regression.

Lazy Learners Vs. Eager Learners

There are two types of learners in machine learning classification: lazy and eager learners.

Eager learners are machine learning algorithms that first build a model from the training dataset before making any prediction on future datasets. They spend more time during the training process because of their eagerness to have a better generalization during the training from learning the weights, but they require less time to make predictions.

Most machine learning algorithms are eager learners, and below are some examples:

- Logistic Regression.
- Support Vector Machine.
- Decision Trees.
- Artificial Neural Networks.

Lazy learners or instance-based learners, on the other hand, do not create any model immediately from the training data, and this is where the lazy aspect comes from. They just memorize the training data, and each time there is a need to make a prediction, they search for the nearest neighbor from the whole training data, which makes them very slow during prediction. Some examples of this kind are:

- K-Nearest Neighbor.
- Case-based reasoning.

However, some algorithms, such as **BallTrees** and **KDTrees**, can be used to improve the prediction latency.

Machine Learning Classification Vs. Regression

There are four main categories of Machine Learning algorithms: supervised, unsupervised, semi-supervised, and reinforcement learning.

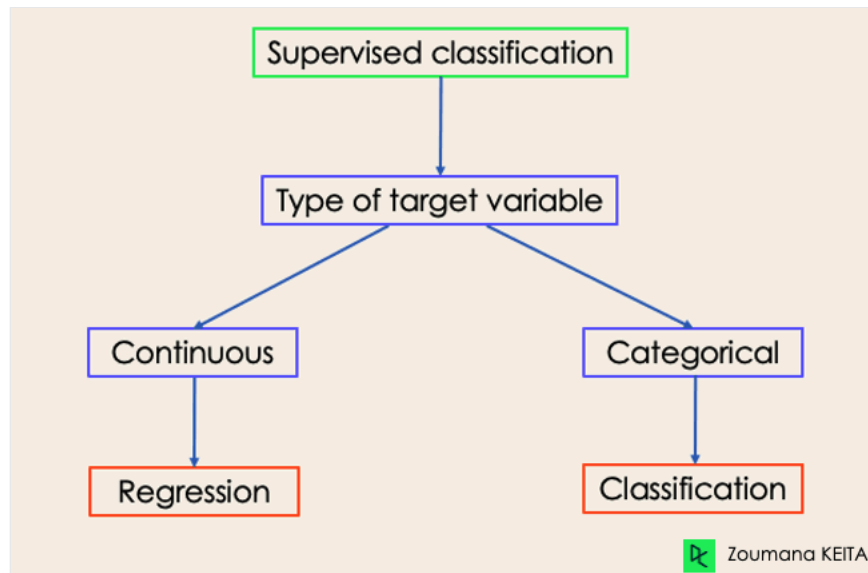
Even though classification and regression are both from the category of supervised learning, they are not the same.

- The prediction task is a **classification** when the target variable is discrete. An application is the identification of the underlying sentiment of a piece of text.

- The prediction task is a **regression** when the target variable is continuous. An example can be the prediction of the salary of a person given their education degree, previous work experience, geographical location, and level of seniority.

If you are interested in knowing more about classification, courses on [Supervised Learning with scikit-learn](#) and [Supervised Learning in R might be helpful](#). They provide you with a better understanding of how each algorithm approaches tasks and the Python and R functions required to implement them.

Regarding regression, [Introduction to Regression in R](#) and [Introduction to Regression with statsmodels in Python](#) will help you explore different types of regression models as well as their implementation in R and Python.



Examples of Machine Learning Classification in Real Life

Supervised Machine Learning Classification has different applications in multiple domains of our day-to-day life. Below are some examples.

Healthcare

Training a machine learning model on historical patient data can help healthcare specialists accurately analyze their diagnoses:

- During the COVID-19 pandemic, machine learning models were implemented to efficiently predict whether a person had COVID-19 or not.
- Researchers can use machine learning models to predict new diseases that are more likely to emerge in the future.

Education

Education is one of the domains dealing with the most textual, video, and audio data. This unstructured information can be analyzed with the help of Natural Language technologies to perform different tasks such as:

- The classification of documents per category.
- Automatic identification of the underlying language of students' documents during their application.
- Analysis of students' feedback sentiments about a Professor.

Transportation

Transportation is the key component of many countries' economic development. As a result, industries are using machine and deep learning models:

- To predict which geographical location will have a rise in traffic volume.
- Predict potential issues that may occur in specific locations due to weather conditions.

Sustainable agriculture

Agriculture is one of the most valuable pillars of human survival. Introducing sustainability can help improve farmers' productivity at a different level without damaging the environment:

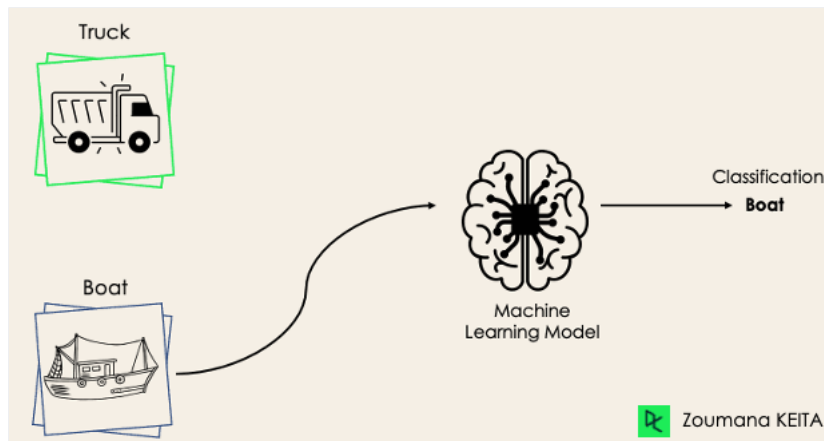
- By using classification models to predict which type of land is suitable for a given type of seed.
- Predict the weather to help them take proper preventive measures.

Different Types of Classification Tasks in Machine Learning

There are four main classification tasks in Machine learning: binary, multi-class, multi-label, and imbalanced classifications.

Binary Classification

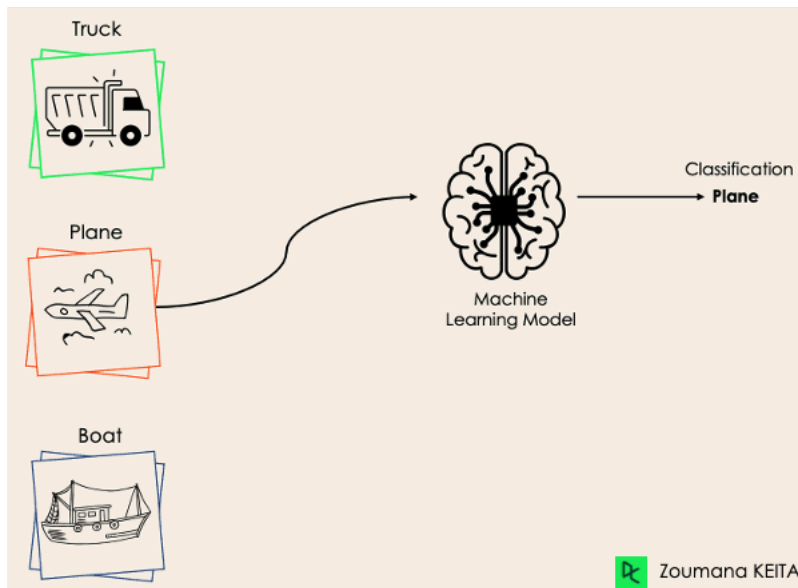
In a binary classification task, the goal is to classify the input data into two mutually exclusive categories. The training data in such a situation is labeled in a binary format: true and false; positive and negative; 0 and 1; spam and not spam, etc. depending on the problem being tackled. For instance, we might want to detect whether a given image is a truck or a boat.



Logistic Regression and Support Vector Machines algorithms are natively designed for binary classifications. However, other algorithms such as K-Nearest Neighbors and Decision Trees can also be used for binary classification.

Multi-Class Classification

The multi-class classification, on the other hand, has at least two mutually exclusive class labels, where the goal is to predict to which class a given input example belongs to. In the following case, the model correctly classified the image to be a plane.



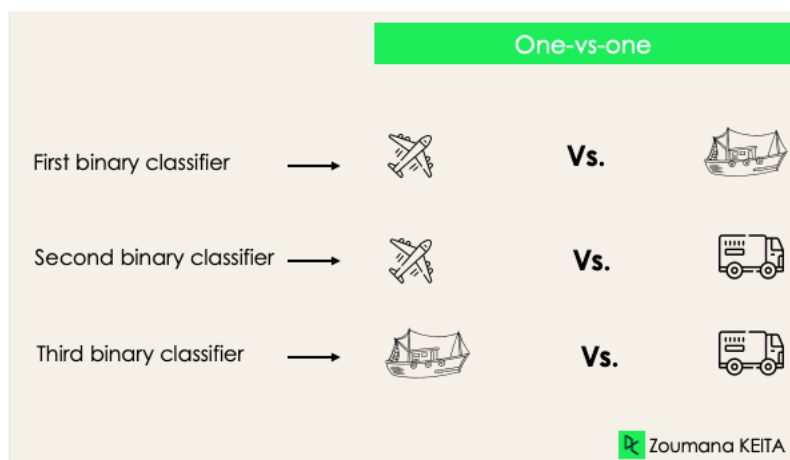
Most of the binary classification algorithms can be also used for multi-class classification. These algorithms include but are not limited to:

- Random Forest
- Naive Bayes
- K-Nearest Neighbors
- Gradient Boosting
- SVM
- Logistic Regression.

But wait! Didn't you say that SVM and Logistic Regression do not support multi-class classification by default?

→ *That's correct. However, we can apply binary transformation approaches such as one-versus-one and one-versus-all to adapt native binary classification algorithms for multi-class classification tasks.*

One-versus-one: this strategy trains as many classifiers as there are pairs of labels. If we have a 3-class classification, we will have three pairs of labels, thus three classifiers, as shown below.

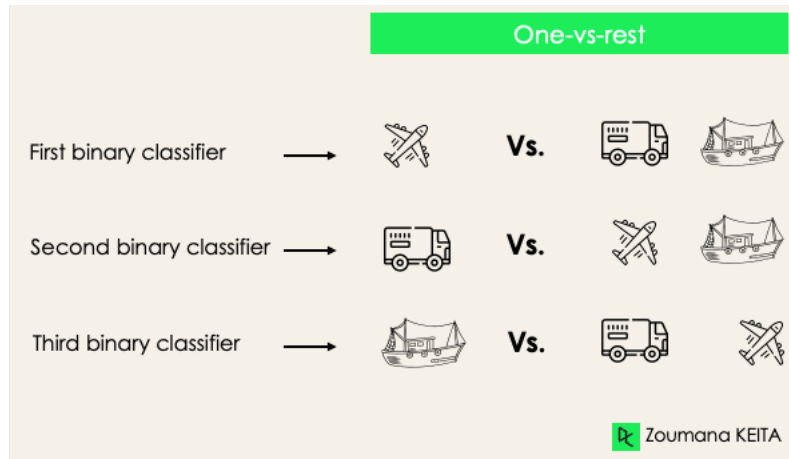


In general, for N labels, we will have $N \times (N-1) / 2$ classifiers. Each classifier is trained on a single binary dataset, and the final class is predicted by a majority vote between all the classifiers. One-vs-one approach works best for SVM and other kernel-based algorithms.

One-versus-rest: at this stage, we start by considering each label as an independent label and consider the rest combined as only one label. With 3-classes, we will have three

classifiers.

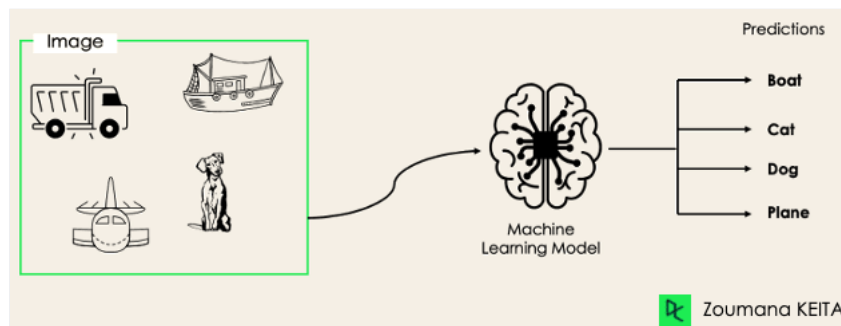
In general, for N labels, we will have N binary classifiers.



Multi-Label Classification

In multi-label classification tasks, we try to predict 0 or more classes for each input example. In this case, there is no mutual exclusion because the input example can have more than one label.

Such a scenario can be observed in different domains, such as auto-tagging in Natural Language Processing, where a given text can contain multiple topics. Similarly to computer vision, an image can contain multiple objects, as illustrated below: the model predicted that the image contains: a plane, a boat, a truck, and a dog.

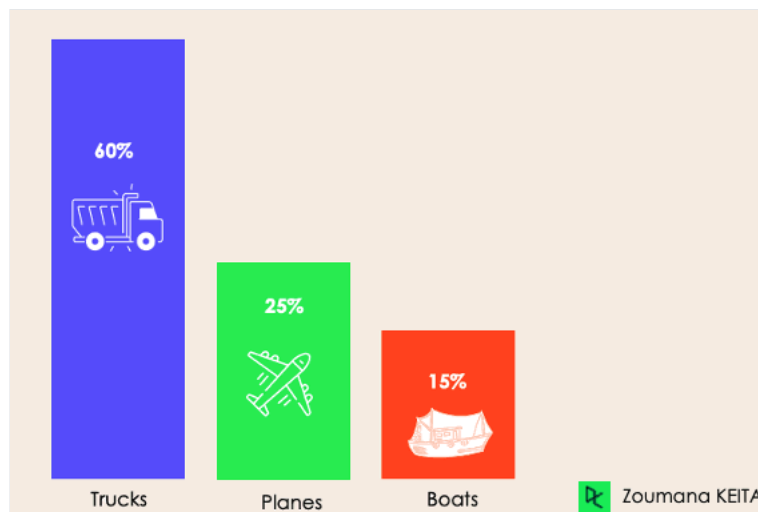


It is not possible to use multi-class or binary classification models to perform multi-label classification. However, most algorithms used for those standard classification tasks have their specialized versions for multi-label classification. We can cite:

- *Multi-label Decision Trees*
- *Multi-label Gradient Boosting*
- *Multi-label Random Forests*

Imbalanced Classification

For the imbalanced classification, the number of examples is unevenly distributed in each class, meaning that we can have more of one class than the others in the training data. Let's consider the following 3-class classification scenario where the training data contains: 60% of trucks, 25% of planes, and 15% of boats.



The imbalanced classification problem could occur in the following scenario:

- *Fraudulent transaction detections in financial industries*
- *Rare disease diagnosis*
- *Customer churn analysis*

Using conventional predictive models such as Decision Trees, Logistic Regression, etc. could not be effective when dealing with an imbalanced dataset, because they might be biased toward predicting the class with the highest number of observations, and considering those with fewer numbers as noise.

So, does that mean that such problems are left behind?

Of course not! We can use multiple approaches to tackle the imbalance problem in a dataset. The most commonly used approaches include sampling techniques or harnessing the power of cost-sensitive algorithms.

Sampling Techniques

These techniques aim to balance the distribution of the original by:

- *Cluster-based Oversampling:*
- *Random undersampling: random elimination of examples from the majority class.*
- *SMOTE Oversampling: random replication of examples from the minority class.*

Cost-Sensitive Algorithms

These algorithms take into consideration the cost of misclassification. They aim to minimize the total cost generated by the models.

- *Cost-sensitive Decision Trees.*
- *Cost-sensitive Logistic Regression.*
- *Cost-sensitive Support Vector Machines.*

Metrics to Evaluate Machine Learning Classification Algorithms

Now that we have an idea of the different types of classification models, it is crucial to choose the right evaluation metrics for those models. In this section, we will cover the most commonly used metrics: accuracy, precision, recall, F1 score, and area under the ROC (Receiver Operating Characteristic) curve and AUC (Area Under the Curve).

A bit of context

Imagine you are a healthcare startup, and want an AI assistant able to predict whether a given patient has a heart disease or not based on its health record. This is a binary classification problem where the model will predict

- 1, True or Yes if the patient has heart disease
- 0, False or No otherwise

1 Confusion matrix

A 2X2 matrix that nicely summarizes the number of correct predictions of the model. It also helps in computing different other performance metrics.

Predict\Reality	Yes	No
Yes	True Positives (TP)	False Negatives (FN)
No	False Positives (FP)	True Negatives (TN)

Type I Error


Type II Error

Type I & II Errors can be used interchangeably when referring to False Positives and False negatives respectively

2 Accuracy

We get accuracy by answering this question: "out of the predictions made by the model, what percentage is correct?"

$$\text{Accuracy} = \frac{TP + TN}{\text{Total number observation}}$$

 Zoumana KEITA
3 Precision

We get precision by answering this question: "out of all the YES predictions, how many of them were correct?"

$$\text{Precision} = \frac{TP}{TP + FP}$$

4 Recall / Sensitivity

It aims to answer this question: "how good was the model at predicting real Yes events?", which can be considered as the flip of the precision.

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

5 Recall / Specificity


It aims to answer this question: "how good was the model at predicting real No events?"

$$\text{Specificity} = \frac{TN}{TN + FP}$$

6 F1 Score

Sometimes used when dealing with imbalanced data set, meaning that there are more of one class/label than there are of the other. It corresponds to the harmonic mean of the precision and recall.

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

 Zoumana KEITA

7 AUC – ROC Curve

AUC– ROC generates probability values instead of binary 0/1 values. It should be used when your data set is roughly balanced.

Using ROC for imbalanced data sets lead to incorrect interpretation.

ROC curves provide good overview of trade-off between the TP rate and FP rate for binary classifier using different probability thresholds.

- A value below 0.5 indicates a poor classifier
- A value of 0.5 means random classifier
- Value over 0.7 corresponds to a good classifier
- 0.8 indicates a strong classifier
- We have 1 when the classifier perfectly predicts everything.

Strategies to choose the right metric**Choose accuracy**

- The cost of FP and FN are roughly equal.
- The benefit of TP and TN are roughly equal.

Choose Precision

- The cost of FP is much higher than a FN.
- The benefit of a TP is much higher than a TN.

Choose recall

- The cost of FN is much higher than a FP.
- The cost of a TN is much higher than a TP.

ROC AUC & Precision – Recall curves

- Use ROC when the dealing with balanced data sets.
- Use precision-recall for imbalanced data sets.



Zoomana KEITA

Deep Dive into Classification Algorithms

We now have all the tools in hand to proceed with the implementation of some algorithms. This section will cover four algorithms and their implementation on the [loans dataset](#) to illustrate some of the previously covered concepts, especially for the imbalanced datasets using a binary classification task. We will focus on only four algorithms for simplicity's sake.

The goal is not to have the best possible model but to illustrate how to train each of the following algorithms. The source code is available on the [DataCamp Workspace](#), where you can execute everything with one click.

Distribution of Loans in the Dataset

- Look at the first five observations in the dataset.

```
import pandas as pd
loan_data = pd.read_csv("loan_data.csv")
loan_data.head()
```



Collapse Visualize		5 rows						
	credit.policy	purpose	int.rate	installment	log.annual.inc	dti	fico	days.wit
0	1	debt_consolidation	0.1189	829.1	11.35040654	19.48	737	
1	1	credit_card	0.1071	228.22	11.08214255	14.29	707	
2	1	debt_consolidation	0.1357	366.86	10.37349118	11.63	682	
3	1	debt_consolidation	0.1008	162.34	11.35040654	8.1	712	
4	1	credit_card	0.1426	102.92	11.29973224	14.97	667	

- Borrowers profile in the dataset.

```
import matplotlib.pyplot as plt
# Helper function for data distribution
# Visualize the proportion of borrowers
def show_loan_distrib(data):
```



```

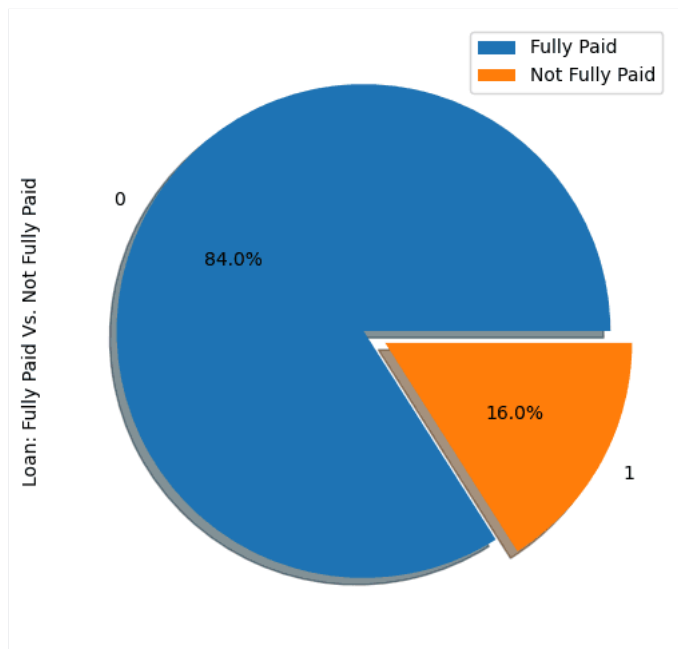
count = ""
if isinstance(data, pd.DataFrame):
    count = data["not.fully.paid"].value_counts()
else:
    count = data.value_counts()

count.plot(kind = 'pie', explode = [0, 0.1],

           figsize = (6, 6), autopct = '%1.1f%%', shadow = True)
plt.ylabel("Loan: Fully Paid Vs. Not Fully Paid")
plt.legend(["Fully Paid", "Not Fully Paid"])
plt.show()

# Visualize the proportion of borrowers
show_loan_distrib(loan_data)

```



From the graphic above, we notice that 84% of the borrowers paid their loans back, and only 16% didn't pay them back, which makes the dataset really imbalanced.

Variable Types

Before further, we need to check the variables' type so that we can encode those that need to be encoded.

We notice that all the columns are continuous variables, except the **purpose** attribute, which needs to be encoded.

```

credit.policy      int64
purpose            object
int.rate           float64
installment        float64
log.annual.inc     float64
dti                float64
fico               int64
days.with.cr.line float64
revol.bal          int64
revol.util         float64
inq.last.6mths     int64
delinq.2yrs        int64
pub.rec            int64
not.fully.paid     int64
dtype: object

```

```
# Check column types
print(loan_data.dtypes)
```



credit.policy	int64
int.rate	float64
installment	float64
log.annual.inc	float64
dti	float64
fico	int64
days.with.cr.line	float64
revol.bal	int64
revol.util	float64
inq.last.6mths	int64
delinq.2yrs	int64
pub.rec	int64
not.fully.paid	int64
purpose_credit_card	uint8
purpose_debt_consolidation	uint8
purpose_educational	uint8
purpose_home_improvement	uint8
purpose_major_purchase	uint8
purpose_small_business	uint8
dtype:	object

Encoding result

```
encoded_loan_data = pd.get_dummies(loan_data, prefix="purpose",
                                   drop_first=True)
print(encoded_loan_data.dtypes)
```



Separate data into train and test

```
X = encoded_loan_data.drop('not.fully.paid', axis = 1)
y = encoded_loan_data['not.fully.paid']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,
                                                    stratify = y, random_state=2022)
```



Buy Now >

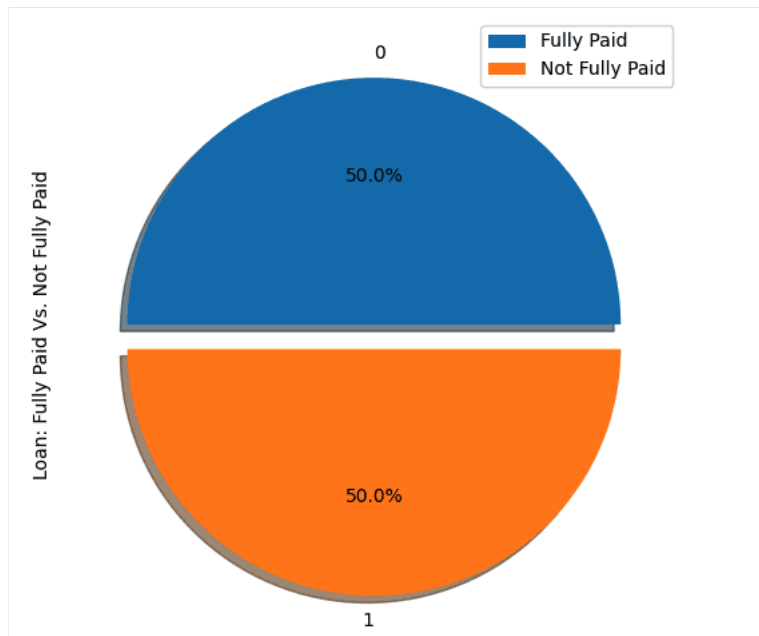
Random Undersampling

We will undersample the majority class, which corresponds to the “fully paid” (class 0).

```
X_train_cp = X_train.copy()
X_train_cp['not.fully.paid'] = y_train
y_0 = X_train_cp[X_train_cp['not.fully.paid'] == 0]
y_1 = X_train_cp[X_train_cp['not.fully.paid'] == 1]
y_0_undersample = y_0.sample(y_1.shape[0])
loan_data_undersample = pd.concat([y_0_undersample, y_1], axis = 0)

# Visualize the proportion of borrowers
show_loan_distrib(loan_data_undersample)
```





SMOTE Oversampling

Perform oversampling on the minority class

```
smote = SMOTE(sampling_strategy='minority')
X_train_SMOTE, y_train_SMOTE = smote.fit_resample(X_train, y_train)
# Visualize the proportion of borrowers
show_loan_distrib(y_train_SMOTE)
```



After applying the sampling strategies, we observe that the dataset is equally distributed across the different types of borrowers.

Application of Some Machine Learning Classification Algorithms

This section will apply these two classification algorithms to the SMOTE smote sampled dataset. The same training approach can be applied to undersampled data as well.

Logistic Regression

This is an explainable algorithm. It classifies a data point by modeling its probability of belonging to a given class using the sigmoid function.

```
X = loan_data_undersample.drop('not.fully.paid', axis = 1)
y = loan_data_undersample['not.fully.paid']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15, stratif
logistic_classifier = LogisticRegression()
logistic_classifier.fit(X_train, y_train)
y_pred = logistic_classifier.predict(X_test)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```



	precision	recall	f1-score	support
0	0.55	0.64	0.59	161
1	0.57	0.48	0.53	161
accuracy			0.56	322
macro avg	0.56	0.56	0.56	322
weighted avg	0.56	0.56	0.56	322

Support Vector Machines

This algorithm can be used for both classification and regression. It learns to draw the hyperplane (decision boundary) by using the margin to maximization principle. This decision boundary is drawn through the two closest support vectors.

SVM provides a transformation strategy called kernel tricks used to project non-linearly separable data onto a higher dimension space to make them linearly separable.

```
from sklearn.svm import SVC
svc_classifier = SVC(kernel='linear')
svc_classifier.fit(X_train, y_train)

# Make Prediction & print the result
y_pred = svc_classifier.predict(X_test)

print(classification_report(y_test, y_pred))
```



	precision	recall	f1-score	support
0	0.59	0.37	0.45	161
1	0.54	0.75	0.63	161
accuracy			0.56	322
macro avg	0.57	0.56	0.54	322
weighted avg	0.57	0.56	0.54	322

These results can be of course improved with more feature engineering and fine-tuning. But they are better than using the original imbalanced data.

XGBoost

This algorithm is an extension of a well-known algorithm called gradient-boosted trees. It is a great candidate not only for combating overfitting but also for speed and performance.

To not make it longer, you can refer to [Machine Learning with Tree-Based Models in Python](#) and [Machine Learning with Tree-Based Models in R](#). From these courses, you will learn how to use both Python and R to implement tree-based models.

Conclusion

This conceptual blog covered the main aspect of classifications in Machine learning and also provided you with some examples of different domains they are applied to. Finally, it covered the implementation of Logistic Regression and Support Vector Machine after performing the undersampling and SMOTE oversampling strategies to generate a balanced dataset for the models' training.

We hope it helped you have a better understanding of this topic of classification in Machine Learning. You can further your learning by following the [Machine Learning Scientist with Python track](#), which covers both supervised, unsupervised and deep learning. It also provides a good introduction to natural language processing, image processing, Spark, and Keras.

Classification FAQs

What are the different types of kernels in SVM? ^

Popular kernels in SVM are Linear Kernel, Polynomial Kernel, Gaussian Kernel, Radial Basis Function (RBF), Laplace RBF Kernel, Sigmoid Kernel, Anova Kernel, Bessel function kernel.

Why do we use classification? v

Which algorithms can be used for both regression and classification? v

Which algorithm is best for multiclass classification? v

Which classification algorithm is best for a small dataset? v

TOPICS

Machine Learning

Machine Learning Courses

COURSE

Supervised Learning in R: Classification

🕒 4 hr 👤 86.5K

In this course you will learn the basics of machine learning for classification.

[See Details →](#)

[Start Course](#)

[See More →](#)

Related



Navigating the World of MLOps
Certifications

Adel Nehme

How to Learn Machine Learning
in 2024

Adel Nehme



A Beginner's Guide to CI/CD for
Machine Learning

Abid Ali Awan

See More →

Grow your data skills with DataCamp for Mobile

Make progress on the go with our mobile courses and daily 5-minute coding challenges.



LEARN

Learn Python

Learn R

Learn AI

Learn SQL

Learn Power BI

Learn Tableau

Learn Data Engineering

Assessments

Career Tracks

Skill Tracks

Courses

Data Science Roadmap

DATA COURSES

Python Courses

R Courses

SQL Courses

Power BI Courses

Tableau Courses

Azure Courses

Spreadsheets Courses

AI Courses

Data Analysis Courses

[Data Visualization Courses](#)

[Machine Learning Courses](#)

[Data Engineering Courses](#)

[Probability & Statistics Courses](#)

WORKSPACE

[Get Started](#)

[Templates](#)

[Integrations](#)

[Documentation](#)

CERTIFICATION

[Certifications](#)

[Data Scientist](#)

[Data Analyst](#)

[Data Engineer](#)

[Hire Data Professionals](#)

RESOURCES

[Resource Center](#)

[Upcoming Events](#)

[Blog](#)

[Code-Alongs](#)

[Tutorials](#)

[Open Source](#)

[RDocumentation](#)

[Course Editor](#)

[Book a Demo with DataCamp for Business](#)

[Data Portfolio](#)

[Portfolio Leaderboard](#)

PLANS

[Pricing](#)

[For Business](#)

[For Universities](#)

[Discounts, Promos & Sales](#)

[DataCamp Donates](#)

SUPPORT

[Help Center](#)

[Become an Affiliate](#)

ABOUT

[About Us](#)

[Learner Stories](#)

[Careers](#)

[Become an Instructor](#)

[Press](#)

[Leadership](#)

[Contact Us](#)

[DataCamp Español](#)



[Privacy Policy](#)

[Cookie Notice](#)

[Do Not Sell My Personal Information](#)

[Accessibility](#)

[Security](#)

[Terms of Use](#)

© 2024 DataCamp, Inc. All Rights Reserved.