Project Name: Handling Imbalanced Dataset

Table of Content

Demo

Cross Validation Like KFOLD and Hyperpaqrameter Tuning

In [5]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.model_selection import KFold
import numpy as np
from sklearn.model_selection import GridSearchCV
```

In [6]:
```
10.0 **np.arange(-2,3)
```

Out[6]:
```
array([1.e-02, 1.e-01, 1.e+00, 1.e+01, 1.e+02])
```

In [7]:
```
log_class=LogisticRegression()
grid={'C':10.0 **np.arange(-2,3),'penalty':['l1','l2']}
cv=KFold(n_splits=5,random_state=None,shuffle=False)
```

In [8]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,train_size=0.7)
```

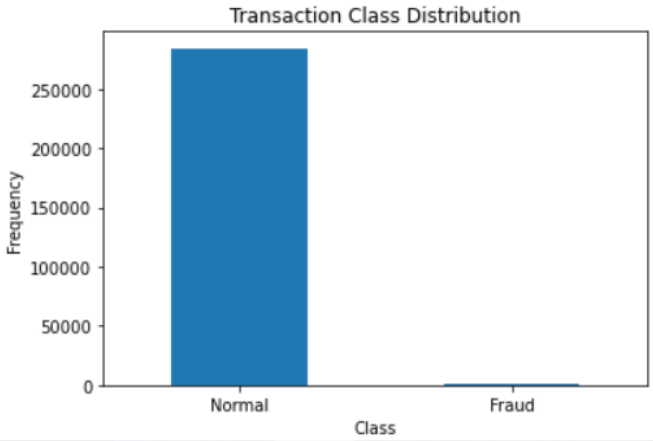
In [9]:
```
clf=GridSearchCV(log_class,grid,cv=cv,n_jobs=-1,scoring='f1_macro')
clf.fit(X_train,y_train)
```

```
C:\Users\Monica\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:762: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[9]:
```
GridSearchCV(cv=KFold(n_splits=5, random_state=None, shuffle=False),
             estimator=LogisticRegression(), n_jobs=-1,
```

In [20]:
```
import matplotlib.pyplot as plt

LABELS = ["Normal", "Fraud"]

count_classes = pd.value_counts(df['Class'], sort = True)

count_classes.plot(kind = 'bar', rot=0)

plt.title("Transaction Class Distribution")

plt.xticks(range(2), LABELS)

plt.xlabel("Class")

plt.ylabel("Frequency")
```

Out[20]: Text(0, 0.5, 'Frequency')



In [40]:
```
y_pred=classifier.predict(X_test)
print(confusion_matrix(y_test,y_pred))
print(accuracy_score(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[85272    15]
 [   36   120]]
0.999403110845827
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     85287
           1       0.89      0.77      0.82       156

    accuracy                           1.00     85443
   macro avg       0.94      0.88      0.91     85443
weighted avg       1.00      1.00      1.00     85443
```

Ensemble Techniques

```
In [41]: from imblearn.ensemble import EasyEnsembleClassifier

In [43]: easy=EasyEnsembleClassifier()
         easy.fit(X_train,y_train)

Out[43]: EasyEnsembleClassifier()

In [45]: y_pred=easy.predict(X_test)
         print(confusion_matrix(y_test,y_pred))
         print(accuracy_score(y_test,y_pred))
         print(classification_report(y_test,y_pred))
```

```
[[82619  2668]
 [   10   146]]
0.9686574675514671
              precision    recall  f1-score   support

           0       1.00      0.97      0.98     85287
           1       0.05      0.94      0.10       156

    accuracy                           0.97     85443
   macro avg       0.53      0.95      0.54     85443
weighted avg       1.00      0.97      0.98     85443
```

Overview

This is a Speech to Text and Text to Speech Converter for performing sentences or phrases.
This repository contains the code for Speech Text Conversion using python's various libraries.
It used Numpy, Pandas, Sklearn and many other libraries.
These libraries help to perform individually one particular transformation.
Numpy is used for working with arrays. It stands for Numerical Python.
Pandas objects rely heavily on Numpy objects.
Sklearn has 100 to 200 models.
The purpose of creating this repository is life is never what you think so similarly data what you get is not what you think.
These python libraries raised knowledge in discovering these libraries with practical use of it.
It leads to growth in my ML repository.
The screenshots will help you to understand flow of output.

Motivation

The reason behind making this is, working with data in real world has never been an easy so I wanted to understand working of Random Under sampling and Oversampling. And at the same time, no loss of useful information is important for companies that they expect from employees. So, this is necessary in my repo. "Imbalanced Classification Problems are Everywhere!" And "Imbalanced Data Handling is Hard!" – hence this is an effort from my end to deal with it.

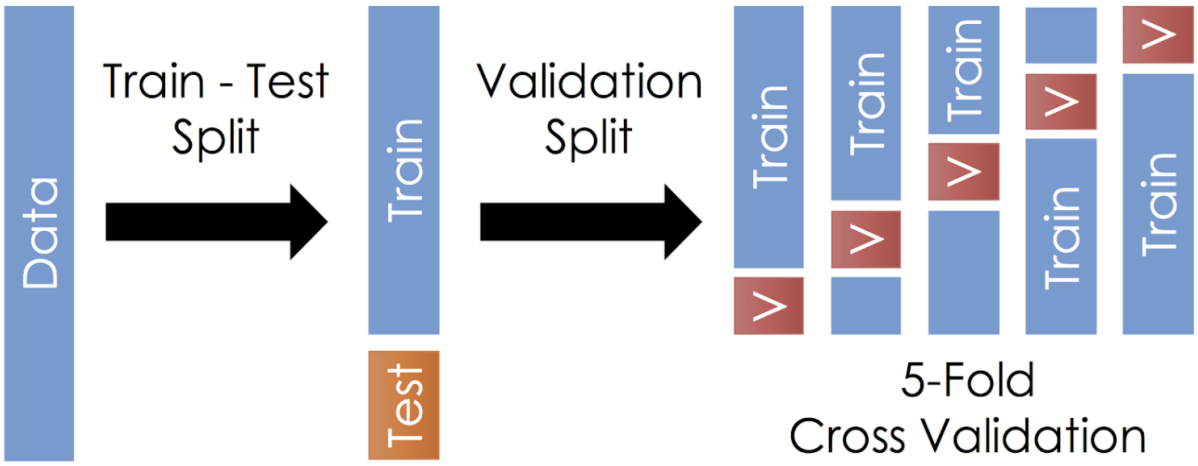
Numpy contains a multi-dimensional array and matrix data structures. It works with the numerical data. Numpy is faster because is densely packed in memory due to its homogeneous type. It also frees the memory faster.
Pandas module mainly works with the tabular data. It contains Data Frame and Series. Pandas is 18 to 20 times slower than Numpy. Pandas is seriously a game changer when it comes to cleaning, transforming, manipulating and analyzing data.
Sklearn is known as scikit learn. It provides many ML libraries and algorithms for it. It provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python.
Logistic regression is appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). I Used simple logistic regression here because when you have one nominal variable and one measurement variable, and you want to know whether variation in the measurement variable causes variation in the nominal variable.
Confusion matrix, accuracy_score, classification_report – sklearn.matrix – Refer to

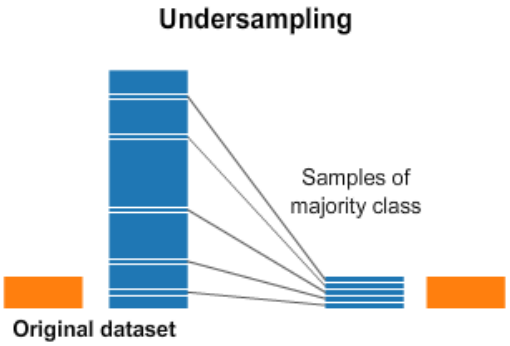
https://scikit-learn.org/stable/modules/model_evaluation.html

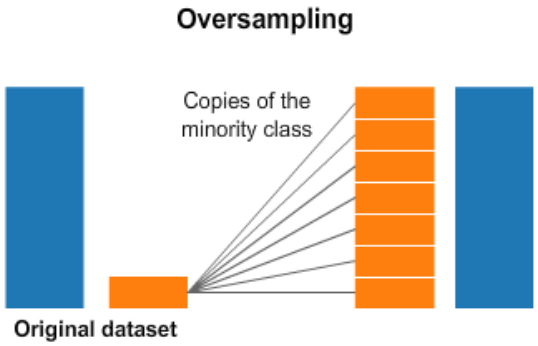
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html

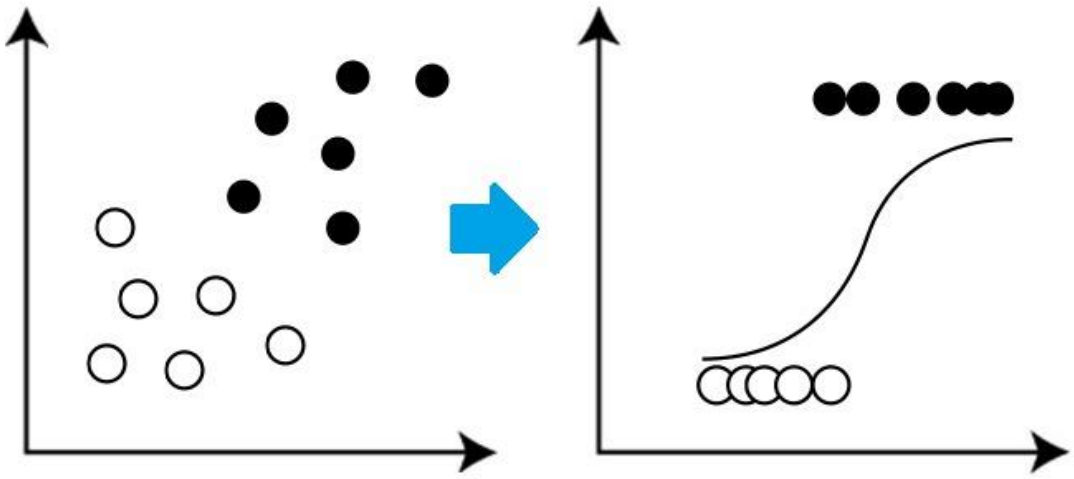
A confusion matrix is a summary of prediction results on a classification problem.
The classification report visualizer displays the precision, recall, F1, and support scores for the model. In order to support easier interpretation and problem detection, the report integrates numerical scores with a color-coded heatmap. All heatmaps are in the range `(0.0, 1.0)` to facilitate easy comparison of classification models across different classification reports.
Kfold, Gridsearchcv – sklearn.model_selection - Cross validation iterators can also be used to directly perform model selection using Grid Search for the optimal hyperparameters of the model. Model_selection is a method for setting a blueprint to analyse data and then using it to measure new data. Selecting a proper model allows you to generate accurate results when making a prediction.
Need to train_test_split - Using the same dataset for both training and testing leaves room for miscalculations, thus increases the chances of inaccurate predictions.
The `train_test_split` function allows you to break a dataset with ease while pursuing an ideal model. Also, keep in mind that your model should not be overfitting or underfitting.
Random forest is a supervised algorithm. It creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting.
Imblearn offers a number of re-sampling techniques commonly used in datasets showing strong between-class imbalance. Most classification algorithms will only perform optimally when the number of samples of each class is roughly the same. Highly skewed datasets, where the minority is heavily outnumbered by one or more classes, have proven to be a challenge. One way of addressing this issue is by re-sampling the dataset as to offset this imbalance with the hope of arriving at a more robust and fair decision boundary than you would otherwise.

The challenge of working with imbalanced datasets is that most machine learning techniques will ignore, and in turn have poor performance on, the minority class, although typically it is performance on the minority class that is most important.
One approach to addressing imbalanced datasets is to oversample the minority class. The simplest approach involves duplicating examples in the minority class, although these examples don't add any new information to the model. Instead, new examples can be synthesized from the existing examples. This is a type of data augmentation for the minority class and is referred to as the Synthetic Minority Oversampling Technique, or SMOTE for short.
SMOTE first selects a minority class instance a at random and finds its k nearest minority class neighbors. The synthetic instance is then created by choosing one of the k nearest neighbours b at random and connecting a and b to form a line segment in the feature space. The synthetic instances are generated as a convex combination of the two chosen instances a and b.










```
[[82619  2668]
 [   10   146]]
0.9686574675514671
              precision    recall  f1-score   support

           0       1.00      0.97      0.98     85287
           1       0.05      0.94      0.10       156

    accuracy                           0.97     85443
   macro avg       0.53      0.95      0.54     85443
weighted avg       1.00      0.97      0.98     85443
```

Installation

Using intel core i5 9th generation with NVIDIA GFORECE GTX1650.
Windows 10 Environment Used.
Already Installed Anaconda Navigator for Python 3.x
The Code is written in Python 3.8.
If you don't have Python installed then please install Anaconda Navigator from its official site.
If you are using a lower version of Python you can upgrade using the pip package, ensuring you have the latest version of pip, *python* -m pip install --*upgrade pip and press Enter*.

Run/How to Use/Steps

Keep your internet connection on while running or accessing files and throughout too.
Follow this when you want to perform from scratch.

Open Anaconda Prompt, Perform the following steps:

cd <PATH>
pip install pandas

pip install numpy
pip install sklearn
pip install matplotlib
pip install imbalanced-learn

You can also create requirement.txt file as, pip freeze > requirements.txt

run files.

Follow this when you want to just perform on local machine.
Download ZIP File.
Right-Click on ZIP file in download section and select Extract file option, which will unzip file.
Move unzip folder to desired folder/location be it D drive or desktop etc.
Open Anaconda Prompt, write cd <PATH> and press Enter.
eg: cd C:\Users\Monica\Desktop\Projects\Python Projects 1\9)Handling_Imbalanced_Data

In Anconda Prompt, pip install -r requirements.txt to install all packages.
Open in Jupyter Notebook, <filename>.ipynb
That is,
Open in Jupyter Notebook, 1)Handling_Imbalanced_Data.ipynb
You can also run all codes from Command Prompt instead of Anaconda Prompt after setting Environmental Variable Path Settings.
Please be careful with spellings or numbers while typing filename and easier is just copy filename and then run it to avoid any silly errors.
Note: cd <PATH>
[Go to Folder where file is. Select the path from top and right-click and select copy option and paste it next to cd one space <path> and press enter, then you can access all files of that folder] [cd means change directory]

Directory Tree/Structure of Project

Folder: 9)Handling_Imbalanced_Data
1)Handling_Imbalanced_Data.ipynb

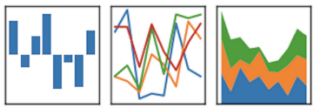
To Do/Future Scope

Can try other technique for same dataset.

I explored 5 different methods for dealing with imbalanced datasets:

1. Change the performance metric.
2. Change the algorithm.
3. Oversample minority class.
4. Under-sample majority class.
5. Generate synthetic samples.

Technologies Used/System Requirement/Tech Stack


Credits

Krish Naik Channel