#NoPayJan Offer - Access all CloudxLab Courses for free between 1st to 31st Jan ENROLL NOW >>

CLOUD x LAB

Feed Home Explore Courses Resources My Courses 5 2,740 XP R

Project - Credit Card Fraud Detection using Machine Learning

100% completed

13 Concepts | 12 Assessments | 213 Learners

Python | Classification | scikit-learn | Logistic Regression | Pandas | SMOTE | Free | Guided Project

Welcome to this project on Credit Card Fraud Detection. In this project, you will use Python, SMOTE Technique(to over-sample data), build a Logistic Regression Classifier, and apply it to detect if a transaction is fraudulent or not.

The real world datasets often might be with data of imbalanced classes. It is very important to feed a decent number of data samples of each class in a classification problem so that the classifier would detect the underlying hidden patterns for each class and prepare itself to reasonably classify the test data. Upon completing this project, you will understand the pragmatic application of various Pandas functions, with a clear picture of how to over-sample the dataset with imbalanced classes using the SMOTE technique and how to use the thus obtained data to train a classifier.

Skills you will develop:

1. Pandas
2. Python Programming
3. SMOTE
4. Scikit-Learn

Attempted by

 Rashmi Ranjan

S Sandeep

 Vignesh

A Akshya

and 209 more

This topic is part of below listed courses -

Artificial Intelligence Deep Learning IIT Roorkee - Batch 1 | Data Science Specialization | Data Science Specialization - EICT, IITR | Artificial Intelligence Deep Learning IIT Roorkee - Batch 2

Instructor:

Vagdevi K

Machine Learning Engineer @ CloudxLab

START NOW

All rights reserved © 2021 CloudxLab Inc

Credit Card Fraud Detection - About the Project

It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase.

The datasets contain transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, ... V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

The dataset has been collected and analyzed during a research collaboration of Worldline and the Machine Learning Group (http://mlg.ulb.ac.be) of ULB (Université Libre de Bruxelles) on big data mining and fraud detection. The dataset could be found in https://www.kaggle.com/mlg-ulb/creditcardfraud

More details on current and past projects on related topics are available on https://www.researchgate.net/project/Fraud-detection-5 and the page of the DefeatFraud project

Mark as Completed

Request Certificate

Index Next

6 Comments

Unfollow conversation

Add comment

This comment has been removed.

Amit Kumar Padhi 3 months ago

hi vagdevi,

Could you please help with guided project on signature verification .

Upvote Reply Share

Rajtilak Bhattacharjee 3 months ago

Hi,

Please check my reply to your mail.

Thanks.

Upvote Reply Share

This comment has been removed.

Hemant Lokras 3 months ago

Sir

Having completed the project , is still showing 96 % complete . Kindly help

rgds

Hemant Lokras

Upvote Reply Share

Vagdevi K 3 months ago

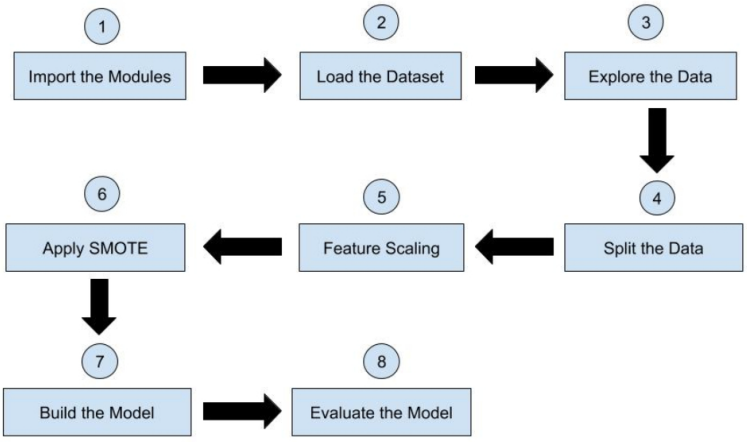
Hi,

Please check if any of the slides are marked with " x " in red color and make sure they are completed.

Thanks.

Upvote Reply Share

Step 0 - Workflow



Mark as Completed

Request Certificate

Previous Index Next

2 Comments

Unfollow conversation

Add comment

Rashid Ahmed 3 months ago

How to learn

Upvote Reply Share

Rajtilak Bhattacharjee 3 months ago

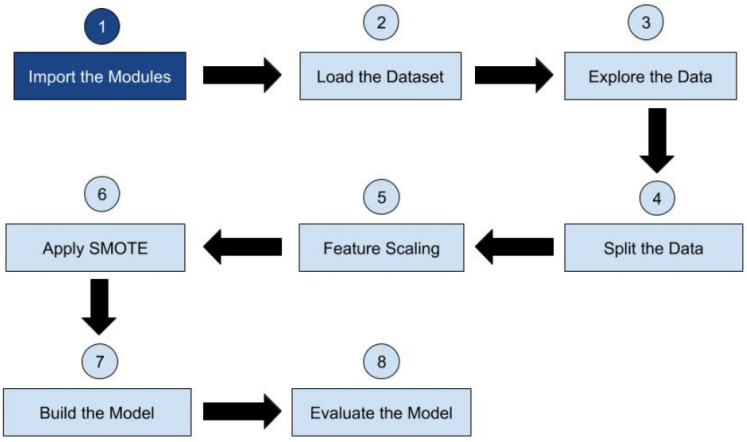
Hi,

This is a guided project which contains steps along with instructions that you need to follow to code. You learn by doing here.

Thanks.

Upvote Reply Share

Step 1 - Import the Modules



Mark as Completed

Request Certificate

Previous | Index | Next

Be the first one to comment!

Add comment

Importing the Modules

We shall begin by importing the necessary modules.

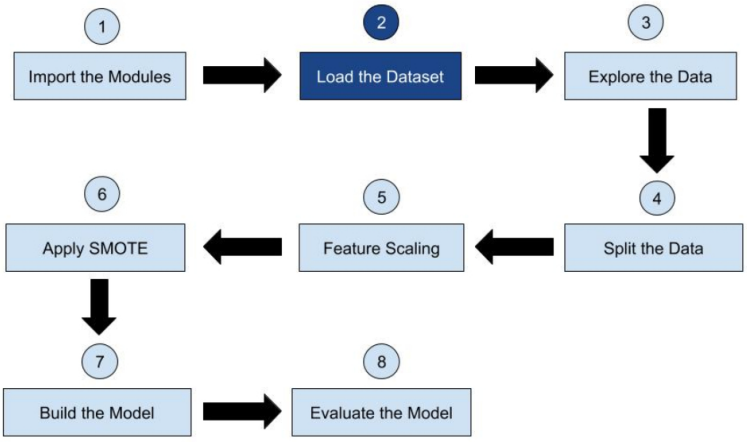
INSTRUCTIONS

- Import the following modules.
  - Import Pandas as pd
  - Import Matplotlib's Pyplot as plt
  - Import Seaborn as sns

Submit Answer | Get Hint | See Answer | Request Certificate

Note - Having trouble with the assessment engine? Follow the steps listed here

Step 2 - Load the Dataset



Mark as Completed
Request Certificate
Previous
Index
Next

Be the first one to comment!

Add comment

Loading the Data

The data is stored in the location `/cxldata/projects/creditcard.csv`.

Let us load the data to our work session and display the top 10 rows.

Note:

`read_csv()` reads a comma-separated values (csv) file into DataFrame.

`head()` displays the top 5 rows of the data frame, whereas `head(n)` displays the top n rows of the data frame.

`shape` of a data frame returns a tuple with the number of rows and columns of the data frame.

INSTRUCTIONS

- Load the data into `data` from the location `/cxldata/projects/creditcard.csv` using `pd.read_csv`.

```
data = << your code comes here >>('/cxldata/projects/creditcard.csv')
```

- Display the top 10 rows stored in the dataframe `data` using `head` method.

```
data.<< your code comes here >>(10)
```

- Display the shape of `data` using `shape`.

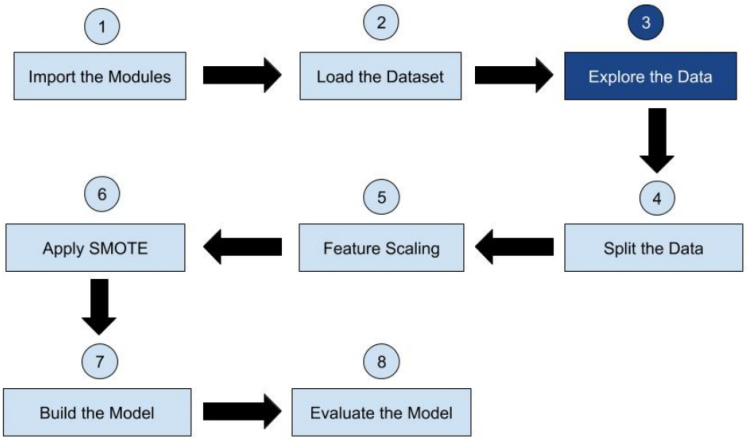
```
data.shape
```

Make sure to write each of these 3 lines in separate code-cells to be able to see the outputs.

Submit Answer | Get Hint | See Answer | Request Certificate

Note - Having trouble with the assessment engine? Follow the steps listed here

Step 3 - Explore the Data



Mark as Completed

Request Certificate

Previous Index Next

Be the first one to comment!

Add comment

Checking for Nulls

The first thing we must do is gather a basic sense of our data. Remember, except for the transaction amount and time columns, we don't know what the other columns are (due to privacy reasons). The only thing we know is that those columns that are unknown have been scaled already.

Let us get the statistical descriptions for each of the numerical columns in the data. We shall also check if there are any null values in our data.

Note:

- `describe()` is a method used on a data frame to view the statistical description of the numerical columns in the data frame.
- `isnull()` method returns True in the places where there are null values(or missing values) and False if the values are not nulls.
- `isnull().sum()` displays column-wise information about the number of nulls found in each column of the data frame.

INSTRUCTIONS

- Use the `describe` method on the data frame `data` to get the statistical description of the data.

```
data.<< your code comes here >>()
```

- Check for nulls in the `data` using `isnull().sum()`.

```
data.<< your code comes here >>()
```

We observe that the data is having no nulls in any of the columns, so we don't have to work on ways to replace values.

Submit Answer | Get Hint | See Answer | Request Certificate

Note - Having trouble with the assessment engine? Follow the steps listed here

Previous | Index | Next

Be the first one to comment!

Add comment

Exploring the Class Column

Let us now explore a bit deeper about the data.

1. Let us first divide the data into features and labels.
2. Then we shall calculate the percentage of the fraud transaction and valid transactions in the dataset and graphically represent the same.

Note:

- `df.loc` of pandas is used to access a group of rows and columns of data frame `df` by label(s) or a boolean array. `.loc[]` is primarily label based, but may also be used with a boolean array.
- `df.value_counts()` of pandas returns a series containing counts of unique values. The resulting object will be in descending order so that the first element is the most frequently-occurring element. Excludes NA values by default.
- `round()` function of Python returns a floating-point number that is a rounded version of the specified number, with the specified number of decimals.

INSTRUCTIONS

- Store all the rows with all columns except the "Class" column into `X`, the feature set.

```
X = data.loc[:, data.columns != 'Class']
```

- Store "Class" values into `Y`, the label set.

```
y = data.loc[:, data.columns == 'Class']
```

- Print the value counts of frauds and non-frauds in the `data` using `value_counts()` on `data['Class']`.

```
print(data['Class'].<< your code comes here >>)
```

Observe, there are more of non-fraud transactions compared to fraudulent transactions. The `value_counts()` method returned them in decreasing order of counts.

- Calculate the percentage of Fraud and Non-fraud transactions.

```
print('Valid Transactions: ', round(data['Class'].value_counts()[0]/len(data) * 100,2), '% of the dataset')

print('Fraudulent Transactions: ', round(data['Class'].value_counts()[1]/len(data) * 100,2), '% of the dataset')
```

We observe that there is a very high class-imbalance.

Submit Answer | Get Hint | See Answer | Request Certificate

Note - Having trouble with the assessment engine? Follow the steps listed here

Previous | Index | Next

3 Comments

Unfollow conversation

Add comment

Padmashree M 3 months ago
what is class imbalance
Upvote Reply Share

Vagdevi K 3 months ago
Hi,
This is a scenario where the number of observations belonging to one class is significantly lower than those belonging to the other classes.
Thanks.
1 Upvote Reply Share

Padmashree M 3 months ago
thank u
Upvote Reply Share

Visualizing the class Imbalance

Let us visualize the class-imbalance using Seaborn `countplot`.

Note:

- `sns.countplot` shows the counts of observations in each categorical bin using bars.

INSTRUCTIONS

- Mention the colors of the bars to be displayed for each class in the count plot.

```
colors = ['blue','red']
```

- Use `sns.countplot` and pass `'Class'`, `data=data` and `palette=colors` as input arguments.

```
<< your code comes here >>('Class', data=data, palette=colors)
```

We observe that the classes are highly imbalanced with most of the transactions are non-fraud.

Submit Answer | Get Hint | See Answer | Request Certificate

Note - Having trouble with the assessment engine? Follow the steps listed here

Previous | Index | Next

2 Comments

Unfollow conversation

Add comment

Mahesh Urkude 3 months ago

Rectify the typo error...add "t" at `sns.countplot`

Use `sns.countplot` and pass `'Class'`, `data=data` and `palette=colors` as input arguments.

Upvote | Reply | Share

Vagdevi K 3 months ago

Corrected, thank you.

Upvote | Reply | Share

Understanding Class-Imbalance

Why don't we want class imbalance?

- From our analysis, we observe there is a lot of imbalance in the classes, with most of the transactions were Non-Fraud (99.83%) of the time, while Fraud transactions occur (0.17%) of the time in the dataframe.
- Using this imbalanced data as such is not a good idea for training a model to classify if a transaction is fraudulent or not.
- This is because, if this imbalanced data is used to train a model, the algorithm does not have a decent amount of fraudulent-data to learn the patterns of fraudulent transactions. Thus, it most probably assumes that every transaction is non-fraudulent(the dominant class of the data).
- This would be a pity because the model naively assumes but doesn't learn/detect the patterns in order to classify.

Any solution?

Yes! To make the dataset balanced, we could either **undersample** or **oversample** it.

- **Under-sampling:** In undersampling, we reduce the dataset such that the number of samples of one class is to that of the other class. But this method has a trade-off with the amount of information lost in the form of the samples removed.
- **Over-sampling:** Next is the oversampling technique. We increase the number of total samples in the dataset by generating the synthetic samples for the minority class in order to achieve the balance between both the classes. 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.

What are we going to do now?

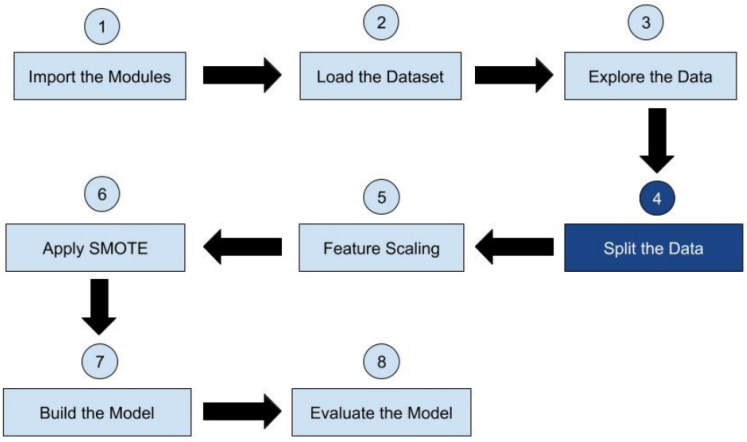
- We should do most pre-processing steps (splitting the data, normalization/standardization, etc) before under/over-sampling the data.
- This is because many sampling techniques require a simple model to be trained (e.g. SMOTE uses a k-NN algorithm to generate samples). These models have better performance on pre-processed datasets (e.g. both k-NN and k-means use euclidean distance, which requires the data to be normalized).
- So, in order for the sampling techniques to work best, we should previously perform any pre-processing steps we can. Then we shall proceed to use SMOTE technique to oversample the train data in order to use it to rain the classification algorithm.

Mark as Completed
Request Certificate
Previous Index Next

Be the first one to comment!

Add comment

Step 4 - Split the Data



Mark as Completed

Request Certificate

Previous Index Next

Be the first one to comment!

Add comment

Splitting the Data

We have to separate the original data frame into train and test sets.

INSTRUCTIONS

- Import the `train_test_split` from `from sklearn.model_selection`

```
from sklearn.model_selection import << your code comes here >>
```

- Split the `X`, `y` into train and test sets using `train_test_split`.

```
X_train, X_test, y_train, y_test =  << your code comes here >>(X, y, test_size=0.3, random_state=0)
```

- Print the shape of the above split-sets.

```
print("Transactions in X_train dataset: ", X_train.shape)
print("Transaction classes in y_train dataset: ", y_train.shape)

print("Transactions in X_test dataset: ", X_test.shape)
print("Transaction classes in y_test dataset: ", y_test.shape)
```

Submit Answer | Get Hint | See Answer

Request Certificate

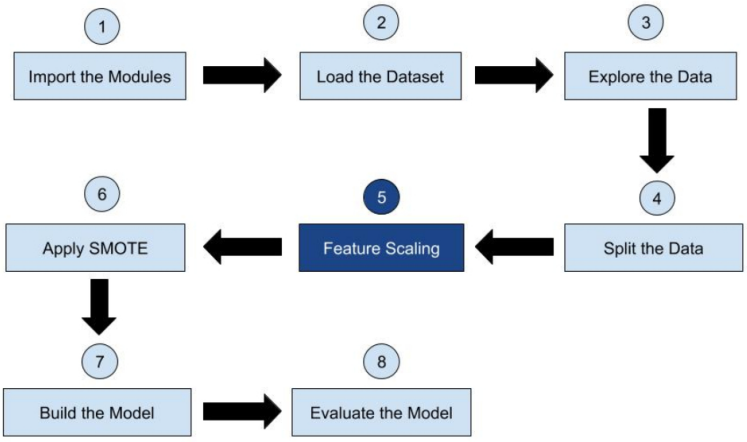
Note - Having trouble with the assessment engine? Follow the steps listed here

Previous | Index | Next

Be the first one to comment!

Add comment

Step 5 - Feature Scaling



Mark as Completed

Request Certificate

Previous Index Next

Be the first one to comment!

Add comment

Feature Scaling

Since most of our data has already been scaled, we should scale the columns that are not yet scaled (Amount and Time).

We shall use StandardScaler to scale the "Amount" column and the "Time" column.

INSTRUCTIONS

- From `sklearn.preprocessing` import `StandardScaler`

```
from sklearn.preprocessing import StandardScaler
```

- Get `StandardScaler()` instances.

```
scaler_amount = StandardScaler()
scaler_time = StandardScaler()
```

- Use `fit_transform` of `scaler_amount` on the `X_train['Amount']` and save the transformed values in `X_train['normAmount']`.

```
X_train['normAmount'] = scaler_amount .<< your code comes here >>(X_train['Amount'].values.reshape(-1, 1))
```

- Use `transform` of `scaler_amount` on the `X_test['Amount']` and save the transformed values in `X_test['normAmount']`.

```
X_test['normAmount'] = scaler_amount .<< your code comes here >>(X_test['Amount'].values.reshape(-1, 1))
```

- Use `fit_transform` of `scaler_time` on the `X_train['Time']` and save the transformed values in `X_train['normTime']`.

```
X_train['normTime'] = scaler_time .<< your code comes here >>(X_train['Time'].values.reshape(-1, 1))
```

- Use `transform` of `scaler_time` on the `X_test['Time']` and save the transformed values in `X_test['normTime']`.

```
X_test['normTime'] = scaler_time .<< your code comes here >>(X_test['Time'].values.reshape(-1, 1))
```

- Drop `Time` and `Amount` columns from `X_train` and `X_test`.

```
X_train = X_train.drop(['Time', 'Amount'], axis=1)
X_test = X_test.drop(['Time', 'Amount'], axis=1)
```

- Display the top 5 rows of `X_train`.

```
X_train.head()
```

Submit Answer | Get Hint | See Answer | Request Certificate

Note - Having trouble with the assessment engine? Follow the steps listed here

Previous | Index | Next

15 Comments

Unfollow conversation

Add comment

Padmashree M 2 months ago

Can u please explain about values.reshape(-1,1)

Upvote Reply Share

Vagdevi K 2 months ago

Hi,

-1 in reshape function is used when you don't know or don't want to explicitly specify the dimension of that axis. For example, if we have an array of shape (2,4) then reshaping it with (-1, 1) would reshape it in such a way that the resulting array has only 1 column. This is only possible by having 8 rows, hence, (8,1).

Hope this helps.

Thanks.

Upvote Reply Share

Padmashree M 3 months ago

why normalization and why are u applying it on time and amount

Upvote Reply Share

Vagdevi K 3 months ago

Hi,

Here we are standardizing the Time and Amount columns. If we take a look at the mean and standard deviations(using `data.describe()`) of all the other features(V1,---,V28), we could observe that the means and stds are all very similar to each other, mimicking the standard normal distribution. But the means and stds of Time and Amount vary greatly when compared to other features. If a feature has a variance that is orders of magnitude larger than others, it might dominate the objective function and make the estimator unable to learn from other features correctly as expected. Thus we are using StandardSclaer to scale the

values of Time and Amount such that their means and stds look similar to those of others.

Hope this helps.

Thanks.

Upvote Reply Share

Bhaswati Bhattacharyya 2 months ago

Thanks for the explanation. Then why dont we normalize the entire data?

Upvote Reply Share

Vagdevi K 2 months ago

Hi,

As already mentioned above, If we take a look at the mean and standard deviations(using `data.describe()`) of all the other features(V1,---,V28), we could observe that the means and stds are all very similar to each other, mimicking the standard normal distribution. But the means and stds of Time and Amount vary greatly when compared to other features. So we normalized the Amount and Time columns only.

Hope this helps.

Thanks.

Upvote Reply Share

Nirzaree Vadgama 3 months ago

I dont feel quite right in trying to scale the time column. I get it from the general technique perspective but aren't there any other implications of this? Can you elaborate a little on the rules on standard scaling here and what if we dont scale the time column?

Upvote Reply Share

Vagdevi K 3 months ago

Hi,

As you could see, the range of values in the time column varies greatly(minimum value: 0, maximum value: 172792.000000, with 84692.000000 mean-value). Unlike the real-world scenario(where we might not feel quite right in trying to scale the time column), all the features are numericals for the neural network. So, if we don't scale them, there is a very high chance for the neural network to give unreasonably large or small importance to the time feature, because of which we might not get the expected output. Hence scaling stands as one of the most important steps.

Hope this helps.

Thanks.

Upvote Reply Share

Nirzaree Vadgama 3 months ago

So we aren't using a NN here. Just logistic regression.

Would be interesting to try the model without the time feature and see if the results are any different.

Or if we were to use the data, use it as a datetime object (for eg.. https://towardsdatascience.com/machine-learning-with-datetime-feature-engineering-predicting-healthcare-appointment-no-shows-5e4ca3a85f96), instead of the kind of scaling done here. Coz we will miss any periodicity of time of day,day of week etc. by the way the data is right now.

Upvote Reply Share

Vagdevi K 3 months ago

Hi,

Basically, we can think of logistic regression as a one-layer neural network.

> Would be interesting to try the model without the time feature and see if the results are any different.

Yes, you could definitely try that out.

> Or if we were to use the data, use it as a datetime object

Yes, periodicity matters, and that would be definitely a very good idea to use datetime object, provided we are given the timestamps. But here in our current dataset, we are provided with the number of seconds elapsed(as mentioned in the description of the dataset). We could get information about the time of day, day of week, etc., only when we have the info about the timestamp but not the elapsed seconds, as we can convert the timestamp into corresponding datetime object (just like as shown in the link you have provided) but it is not the same case with no.of elapsed seconds.

Hope this helps.

Thanks.

Upvote Reply Share

Nirzaree Vadgama 2 months ago

Hey so I tried the model without the time column and the results are not very different between the models (recall stayed the same, only 20 FP instances increased).

Attached : 1. Confusion matrix of model with the time column

2. Confusion matrix of model without time column.

This brings me to a more general question of how do we make the model cut down on features not required? Do we need a separate feature selection step?

Is there a way to incrementally include features like in stepwise decision trees or something else to make sure only the best and sufficient features from all available ones are selected in the model?

Upvote Reply Share

Vagdevi K 2 months ago

Hi,

That's a good observation and glad that you reached out to us with your observation!

>This brings me to a more general question of how do we make the model cut down on features not required? Do we need a separate feature selection step?

We can have a look at the feature importance with respect to a given model. For algorithms like logistic regression, where the prediction model is the weighted sum of input features, we

can directly use those coefficients of the inputs(the weights of the input features learned through the process of training) as their corresponding values of feature importance, which thus could be used to select the features. For example, in our current project, we could access the coefficients of our model using `**lr_gridcv_best.coef_[0]**` which returns the weights learned for each input feature. And for tree-based algorithms like decision trees, random forests etc, the sci-kit learn package readily provides the attribute `**feature_importances_**` when fitted.

> Is there a way to incrementally include features like in stepwise decision trees or something else to make sure only the best and sufficient features from all available ones are selected in the model?

So in ML, we can't generalize something to be best, like a one-size-fits-all solution. It often changes based on the scenario where we want to build the model for. For example, here it is more important for us to make sure no fraudulent transaction is mistakenly classified as a non-fraudulent transaction because this is a monetary related scenario where security should be the at-most priority and thus we can't afford false negatives. So we focussed on recall. In some other situations, like spam email detection, it's ok to classify a spam-email(positive) as a non-spam-email(negative), but it's not ok to mark a non-spam-email as spam, as the user might miss-out some valuable information in a good email. So here we can't afford false positives, and hence precision matters here. JFYI, precision=(TP)/(TP+FP), and recall=(TP)/(TP+FN). So here, we care for high precision, whereas in our fraudulent detection case we care for high recall. So based on our necessity, we generally choose the features which positively affect the higher performance in terms of the choosen metric.

Hope this helps.

Thanks.

1 Upvote Reply Share

Punit Bhilota 3 months ago

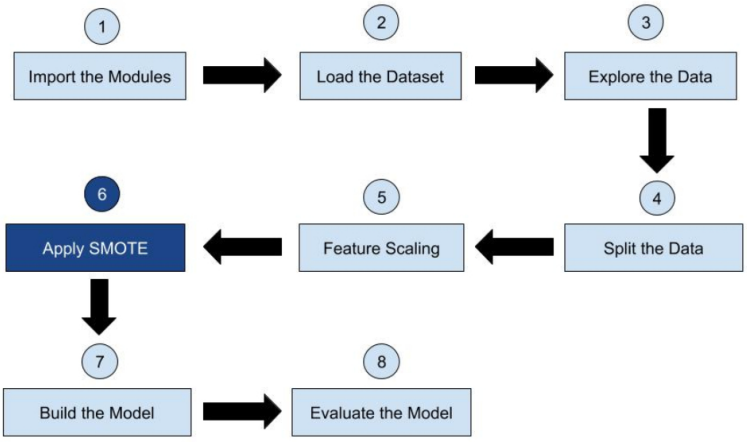
Hello,

Following syntax is not working: I checked the answer as well. Both the syntax are identical.

X_train['normAmount'] = scaler_amount.fit_transform(X_train['Amount'].values.reshape(-1, 1))

```
X_train['normAmount'] = scaler_amount .<< your code comes here >>(X_train['Amount'].values.reshape(-1, 1))
```

Step 6 - Apply SMOTE Technique



Mark as Completed
Request Certificate
Previous
Index
Next

Be the first one to comment!

Add comment

Understanding SMOTE

SMOTE stands for Synthetic Minority Over-sampling Technique. SMOTE creates new synthetic points in order to have an equal balance of the classes. This is another alternative for solving the "class imbalance problems".

Understanding SMOTE:

- **Achieving Balanced Classes :** Using the distances between the closest neighbors of the minority class, SMOTE creates synthetic points in between these distances in order to reach an equal balance between the minority and majority class.
- **Effect:** More information is retained since we don't have to delete any rows unlike in random undersampling(where we remove some data samples of majority class to achieve class balance).
- **Accuracy - Time Tradeoff:** More the number of data samples, more the training time which tends to increase performance. Less the number of data samples, less the information and thus less probable of decent performance.

Mark as Completed
Request Certificate
Previous Index Next

Be the first one to comment!

R Add comment

Applying SMOTE technique

We shall apply the SMOTE technique only on train data and keep the test data untouched so as to avoid any form of data leakage.

Note:

- `imblearn.over_sampling.SMOTE` : Class to perform over-sampling using SMOTE.
- `fit_sample` : This method of `imblearn.over_sampling.SMOTE` is used to resample the dataset.

INSTRUCTIONS

- From `imblearn.over_sampling` import `SMOTE`.

```
from imblearn.over_sampling import SMOTE
```

- Print the number of class-wise samples before over-sampling using the `value_counts` method on `y_train`.

```
print("Before over-sampling:\n", y_train['Class'].<< your code comes here >>)
```

- Declare an instance of `SMOTE` as `sm`.

```
sm = SMOTE()
```

- Use `fit_sample` method of `sm` on `X_train` and `y_train['Class']` and store the resampled features and labels in `X_train_res` and `y_train_res` respectively.

```
X_train_res, y_train_res = sm.<< your code comes here >>(X_train, y_train['Class'])
```

- Print the number of class-wise samples after over-sampling using the `value_counts` method on `y_train`.

```
print("After over-sampling:\n", y_train_res.<< your code comes here >>)
```

Submit Answer | Get Hint | See Answer | Request Certificate

Note - Having trouble with the assessment engine? Follow the steps listed here

Previous | Index | Next

24 Comments

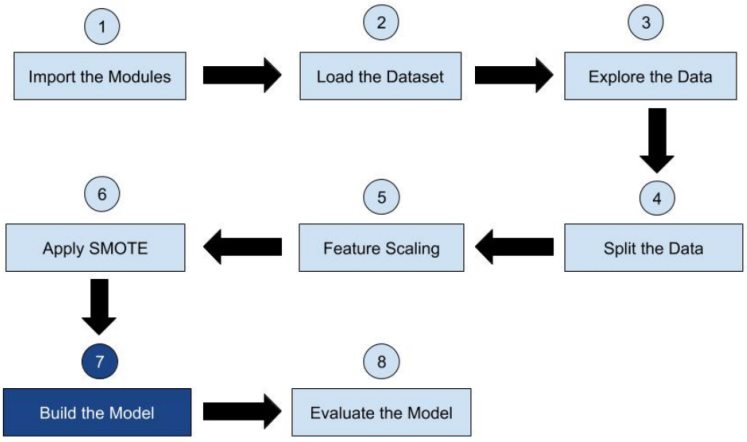
Unfollow conversation

Add comment

Sushovan Chaudhury 2 months ago

```
File "<ipython-input-61-a2abe45ca9be>", line 1
    print("Before over-sampling:\n", y_train['Class'].value_counts()
                                                                    ^
```

Step 7 - Build the Model



Mark as Completed

Request Certificate

Previous Index Next

Be the first one to comment!

Add comment

Training the Classification Algorithm

Let us use logistic regression for this classification problem.

INSTRUCTIONS

- Import `GridSearchCV` from `sklearn.model_selection`.

```
from sklearn.model_selection import << your code comes here >>
```

- Import `LogisticRegression` from `sklearn.linear_model`.

```
from sklearn.linear_model import << your code comes here >>
```

- Import `confusion_matrix, auc, roc_curve` from `sklearn.metrics`.

```
from sklearn.metrics import << your code comes here >>
```

- Let us declare some parameters and their values for the grid-search.

```
parameters = {"penalty": ['l1', 'l2'], 'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000]}
```

paramter C is the regularization parameter, and penalty is the norm used in the penalization.

- Instantiate `LogisticRegression()` as `lr`.

```
lr = LogisticRegression()
```

- Pass `lr, parameters,cv=5` as arguments to `GridSearchCV`.

```
clf = GridSearchCV(lr, parameters, cv=5, verbose=5, n_jobs=3)
```

- Fit the classifier on `X_train_res` and `y_train_res` using `fit`.

```
k = clf.<< your code comes here >>(X_train_res, y_train_res)
```

- Let us print the best parameters.

```
print(k.best_params_)
```

Submit Answer | Get Hint | See Answer | Request Certificate

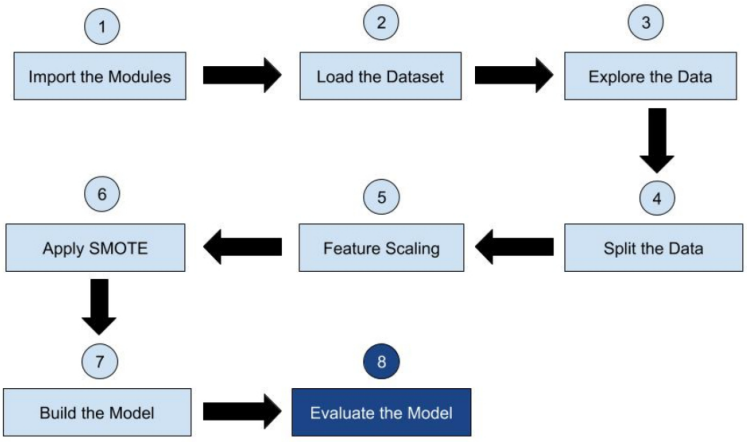
Note - Having trouble with the assessment engine? Follow the steps listed here

Previous | Index | Next

Be the first one to comment!

Add comment

Step 8 - Evaluate the Model



Mark as Completed

Request Certificate

Previous Index Next

Be the first one to comment!

Add comment

Get Confusion matrix and Recall

Let us predict the labels for train and test data, get the confusion matrix, and calculate the recall values.

Note:

- `confusion_matrix` : computes confusion matrix to evaluate the accuracy of classification.
  - By definition, a confusion matrix *C* is such that *Ci,j* is equal to the number of observations known to be in the group *i* and predicted to be in group *j*.
  - Thus in binary classification, the count of true negatives is *C00*, false negatives is *C10*, true positives is *C11* and false positives is *C01*.
- `recall` is calculated by (true positives)/(true positives + false negatives). Note that we are calculating recall value because we want to detect fraudulent credit card transactions. It might be tolerable to classify some valid transactions as fraudulent, but it is not tolerable to misclassify the fraudulent transactions as valid ones.

INSTRUCTIONS

- Store the best estimator from the gridsearchcv in `lr_gridcv_best`.

```
lr_gridcv_best = clf.best_estimator_
```

- Use `predict` method of `lr_gridcv_best` on `X_test` and store the predictions in `y_test_pre`.

```
y_test_pre = lr_gridcv_best.<< your code comes here >>(X_test)
```

- Call the `confusion_matrix` function imported from `sklearn.metrics`. Pass `y_test, y_test_pre` as arguments.

```
cnf_matrix_test = << your code comes here >>(y_test, y_test_pre)
```

- Calculate the recall for test data predictions by the best model.

```
print("Recall metric in the test dataset:", (cnf_matrix_test[1,1]/(cnf_matrix_test[1,0]+cnf_matrix_test[1,1] )))
```

- Use `predict` method of `lr_gridcv_best` on `X_train_res` and store the predictions in `y_train_pre`.

```
y_train_pre = lr_gridcv_best.<< your code comes here >>(X_train_res)
```

- Call the `confusion_matrix` function imported from `sklearn.metrics`. Pass `y_train_res, y_train_pre` as arguments.

```
cnf_matrix_train = << your code comes here >>(y_train_res, y_train_pre)
```

- Calculate the recall for resampled train data predictions by the best model.

```
print("Recall metric in the train dataset:", (cnf_matrix_train[1,1]/(cnf_matrix_train[1,0]+cnf_matrix_train[1,1] )))
```

Submit Answer Get Hint See Answer Request Certificate

Note - Having trouble with the assessment engine? Follow the steps listed here

Previous Index Next

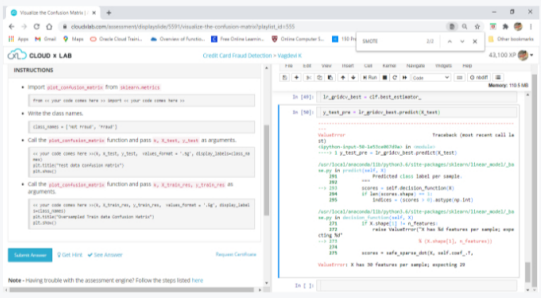
2 Comments

Unfollow conversation

Add comment

Manjari . 3 months ago

I am getting the below error, please help.



Upvote Reply Share

Rajtilak Bhattacharjee 3 months ago

Hi,

The shape of the dataset is incorrect. Would request you to review the code from step 1 against the instructions given, if required please take a hint or look at the answer.

The shape of the dataset is incorrect. Would request you to review the code from step 1 against the instructions given, if required please take a hint or look at the answer.

Thanks.

Upvote Reply Share

Visualize the Confusion Matrix

Let us visualize the confusion matrices for the predictions made on the test set and over-sampled train set.

Note:

`plot_confusion_matrix(estimator, X, y)` plots Confusion Matrix. Here `estimator` is the fitted classifier, `X` is the input values and `y` is the target values.

INSTRUCTIONS

- Import `plot_confusion_matrix` from `sklearn.metrics`

```
from << your code comes here >> import << your code comes here >>
```

- Write the class names.

```
class_names = ['Not Fraud', 'Fraud']
```

- Call the `plot_confusion_matrix` function and pass `k, X_test, y_test` as arguments.

```
<< your code comes here >>(k, X_test, y_test,  values_format = '.5g', display_labels=class_names)
plt.title("Test data Confusion Matrix")
plt.show()
```

- Call the `plot_confusion_matrix` function and pass `k, X_train_res, y_train_res` as arguments.

```
<< your code comes here >>(k, X_train_res, y_train_res,  values_format = '.5g', display_labels=class_names)
plt.title("Oversampled Train data Confusion Matrix")
plt.show()
```

Submit Answer | Get Hint | See Answer | Request Certificate

Note - Having trouble with the assessment engine? Follow the steps listed here

Previous | Index | Next

8 Comments

Unfollow conversation

Add comment

Chaitanya Uppuluri 2 months ago
What is the difference between k and clf in GridSearchCV? Somewhere k is used and clf in another.
1 Upvote | Reply | Share

Vagdevi K 2 months ago
Hi,
They both are the same. We used them just to make the flow clear. Hope this helps.
Thanks.
Upvote | Reply | Share

Prachi Agarwal 3 months ago
Thanks a lot CLOUD X LAB appreciate the efforts
Upvote | Reply | Share

Nirzaree Vadgama 3 months ago
good notebook. Could you also add a step or two about interpreting the model? Features that were important, some intuition on the model.Thanks.
1 Upvote | Reply | Share

Rajtilak Bhattacharjee 3 months ago
Hi,
Thank you for your feedback. We will consider these while updating our projects.
Thanks.
1 Upvote | Reply | Share

Nirzaree Vadgama 3 months ago
Great! Thanks.
Upvote | Reply | Share

Punit Bhilota 3 months ago
values_format = '.6g' instead of '.5g' for confusion matrix plot of train data will enhance the readability.

1 Upvote Reply Share

Rajtilak Bhattacharjee 3 months ago

Hi,

Thank you for your feedback.

Thanks.

Upvote Reply Share

ROC-AUC Curve

Let us now plot the ROC-AUC curve. The Receiver Operator Characteristic (ROC) curve is an evaluation metric for binary classification problems. It is a probability curve that plots the TPR against FPR at various threshold values. The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve. The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.

Note:

- `decision_function` predicts confidence scores for samples. The confidence score for a sample is the signed distance of that sample to the hyperplane. The advantage of Decision Function output is to set DECISION THRESHOLD and predict a new output for X_test, such that we get the desired precision or recall value.
- `roc_curve` computes ROC by taking true binary labels and confidence values, or non-thresholded measure of decisions as input arguments. It returns
  - increasing false-positive rates such that element i is the false positive rate of predictions with score >= thresholds[i] (fpr)
  - Increasing true positive rates such that element i is the true positive rate of predictions with score >= thresholds[i] (tpr)
  - Decreasing thresholds on the decision function used to compute fpr and tpr.

INSTRUCTIONS

- Use `decision_function` method of model `k` and pass `X_test` as argument. Receive the resultant scores in `y_k`.

```
y_k =  k.<< your code comes here >>(X_test)
```

- Call `roc_curve` function by passing `y_test, y_k` as input arguments and receive the returned `fpr`, `tpr` and `thresholds`.

```
fpr, tpr, thresholds = << your code comes here >>(y_test, y_k)
```

- Calculate the Area Under Curve for the `fpr` and `tpr` returned by roc_curve. Call `auc` function.

```
roc_auc = << your code comes here >>(fpr, tpr)
```

- Print the `roc_auc` measure.

```
print("ROC-AUC:", roc_auc)
```

- Now visualize the roc_auc curve.

```
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b',label='AUC = %0.3f'% roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([-0.1,1.0])
plt.ylim([-0.1,1.01])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

Submit Answer | Get Hint | See Answer | Request Certificate

Note - Having trouble with the assessment engine? Follow the steps listed here

Previous | Index | Next

2 Comments

Unfollow conversation

Add comment

Surendra Kumar 3 months ago

y_k = k.decision_function(X_test)

fpr, tpr, thresholds = roc_curve(y_test, y_k)

Summary

- We have been given the Europe credit-card transaction data of 2 days. For privacy reasons, the personal details have been represented in the form of Principle Components. The Amount(the transaction Amount) and Time(the seconds elapsed between each transaction and the first transaction in the dataset) are also part of the columns other than the principal components. The transactions are of valid and fraudulent types. The goal is to build a classifier to detect fraudulent transactions.
- We have first loaded the data, explored it, and checked for any null values. While exploring, we found that the data is of high class-imbalance, with around 99.83% being valid transactions whereas about 0.17% are fraudulent.
- It is not a good idea to train a classifier with such highly imbalanced data as it leads to mere assumptions rather than learning by the algorithm. We could either undersample or oversample the data to achieve a balance between the class-wise data samples.
- We have split the data into train and test parts, in order to prevent any data leakage and to keep the test data untouched, before oversampling.
- We have scaled the Amount and Time features using StandardScaler.
- We then applied the SMOTE technique to oversample the train data and formed a new dataset with the thus obtained over-sampled datapoints.
- We used the GridSearch method with different parameter values, trained logistic regression classifiers with the different combinations of these parameters, and got the best logistic regression classifier which yields the least loss on the over-sampled data-set. All this mechanism is internally implemented by GridSearchCV of sklearn.
- We then used the best estimator thus obtained to evaluate its performance on the unseen test data. We calculated the recall, confusion-matrix and roc-auc scores.

Mark as Completed

Request Certificate

Previous Index

6 Comments

Unfollow conversation

Add comment

Bharti Advani 2 months ago

Well guided project, Thank you.

Upvote Reply Share

Aseem Anand 3 months ago

Hi,

I am getting wrong name on certificate. Is there a way to rectify it?

Upvote Reply Share

Karthik Pawar 3 months ago

Hi Aseem,

Yes, you can correct your name on the certificate.
We will update your name on the certificate to your current first name and last name.

Thank you.

Upvote Reply Share

Punit Bhilota 3 months ago

Hi Vagdevi,

Very well instructed. Concept of SMOTE is explained nicely. The explation on recall, consusion_matrix, roc_auc was helpful.

1 Upvote Reply Share

Rajtilak Bhattacharjee 3 months ago

Hi Punit,

Thank you for your feedback.

Thanks.

Upvote Reply Share

This comment has been removed.