APR 16, 2020 / GEORSARA1 / LEAVE A COMMENT

Data Shift in Machine Learning: what is it and how to detect it

The problem

Lets suppose that John works as a Data Scientist for a Bank. His manager tasks him with creating a model for predicting Probability of Default for home loans. John has some intuition on what input variables he needs and what the output variable is. He reaches the Data Engineer, Nicky, and asks her for the data. Sure enough, he gets the required data, performs his Exploratory Data Analysis and then selects a proper classification algorithm to apply.

Cutting the long story short, John has now developed a fancy Machine Learning model to predict Probability of Default for any given client. Maybe he used a knew to reduce the feature space and the model variance. He cross-validated his model within the training data and he also tested the performance of the model in a hold-out test set. He was able to achieve a good model performance of 80% accuracy in the binary prediction. Now, accuracy is probably the worse metric to check – in reality such an institution would care more about Recall but lets lets stick with accuracy for the sake of simplicity. He presents these findings and results with a fancy barchart to his manager, his manager is really happy and presents the results in higher management, the project gets green light for deployment.

So far so good. But all this happened at November 2019. The model was deployed at that time and worked well for a couple of months. But then it started deteriorating. Little by little, month by month. In March 2020 the model's accuracy dropped to 60% percent. John could not deliver what he had promised. So what happened? Why did the model perform well at start but deteriorated heavily in the long term?

Data shift

As the its name suggests, a data shift occurs when there is a change in the data distribution. When building a Machine Learning model, one tries to unearth the (possibly non-linear) relations between the input and the target variable. Upon creating a model on this data, he then might feed new data **of the same** the input dataset or b) the target variable or c) the underlying patterns and relations betweeen the input and output data. Each one of these situations has a distinct name in Data Science but they all lead to the same thing: model performance degradation.

> *Data shift or data drift, concept shift, changing environments, data fractures are all similar terms that describe the same phenomenon: the different distribution of data between train and test sets*

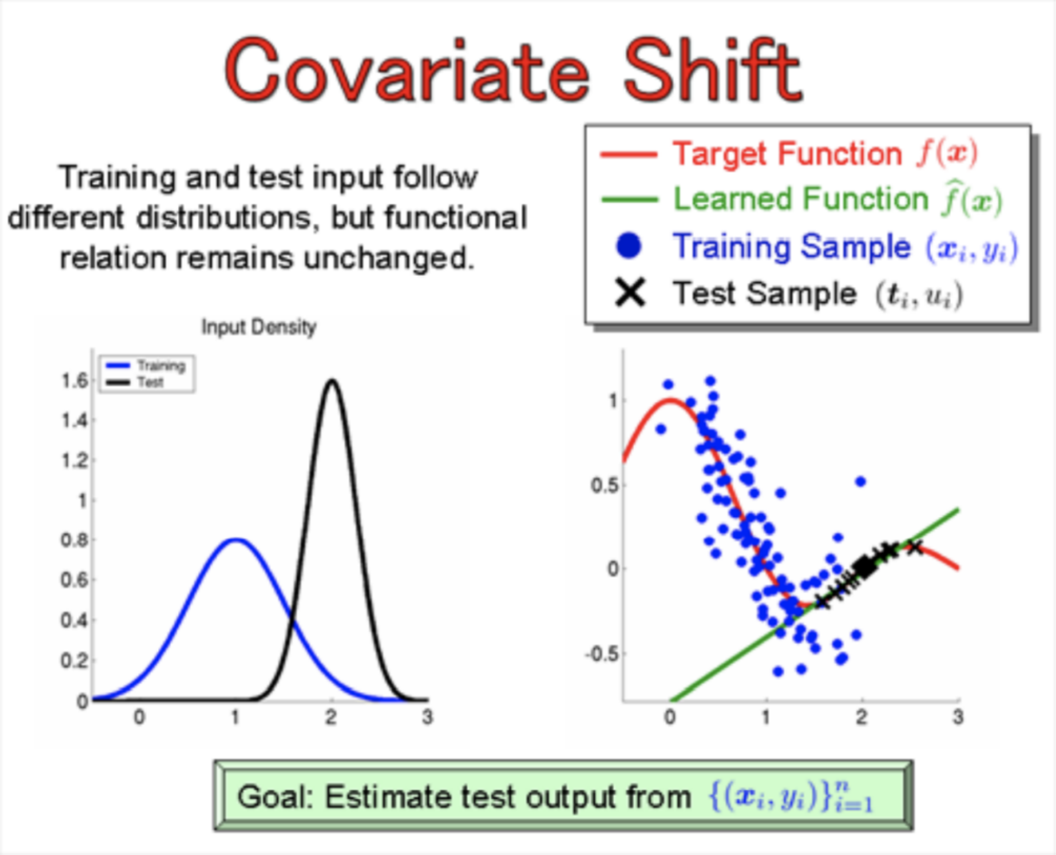
So, in John's case, what really happened is that in the next few months after deploying his model, a very unpredictable thing happened: a global pandemic due to a deadly new virus forced his country's government to impose a citizen lockdown, temporarily shutting down enterprises and heavily reducing economic activity. These major changes affected the behavior of the Bank's clients in repaying their loans: either they **could not do so** because their revenue streams were reduced or **did not want to** because the government granted a 3-month grace period to loan repayments. So what type of data shift did John face? Type (a)? Type (b)? Maybe (c)?

Lets give a more formal definition of each type of data shift and then we will discuss further on John's problem. Moreover, we will see what he could do to **proactively** make sure that a data shift is not present in the data and what to do **after** a data shift is identified.

Formal definitions

1. Covariate shift

Definition 1. Covariate shift is termed the situation where $P_{trn}(Y|X)=P_{tst}(Y|X)$ but $P_{trn}(X) \neq P_{tst}(X)$



Covariate Shift

Training and test input follow different distributions, but functional relation remains unchanged.

Goal: Estimate test output from $\{(x_i, y_i)\}_{i=1}^{n}$

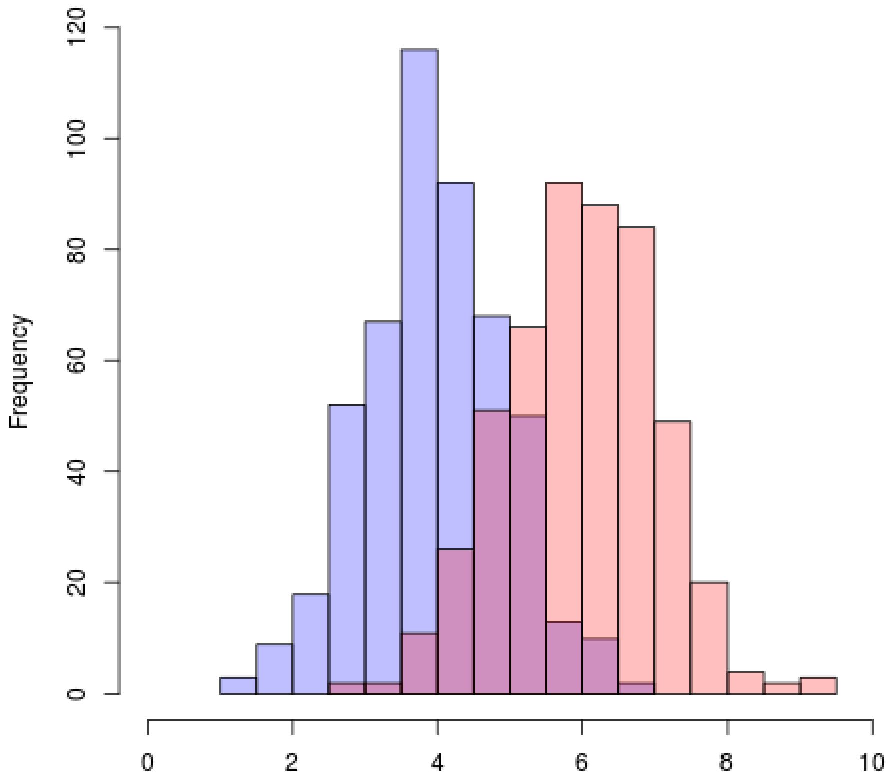
Covariate shift illustration. Source: [1]

Covariate shift may happen due to a changing environment that affects the input variables but not the target variable. In our Probability of Default example with Data Scientist John, this could mean that due to the pandemic many businesses closed or their revenues decreased, their employees became less, etc, however they decided to keep paying their loans because they were affraid that the bank may take their houses (different distributions for the X variables but the same distribution of Y).

Lets proceed with the other two cases.

Prior probability shift can be thought of as the exact opposite of covariate shift: it is the case that input feature distributions remain the same but the distribution of the target variable changes.

Definition 2: Prior probability shift is termed the situation where $P_{trn}(X|Y)=P_{tst}(X|Y)$ but $P_{trn}(Y) \neq P_{tst}(Y)$



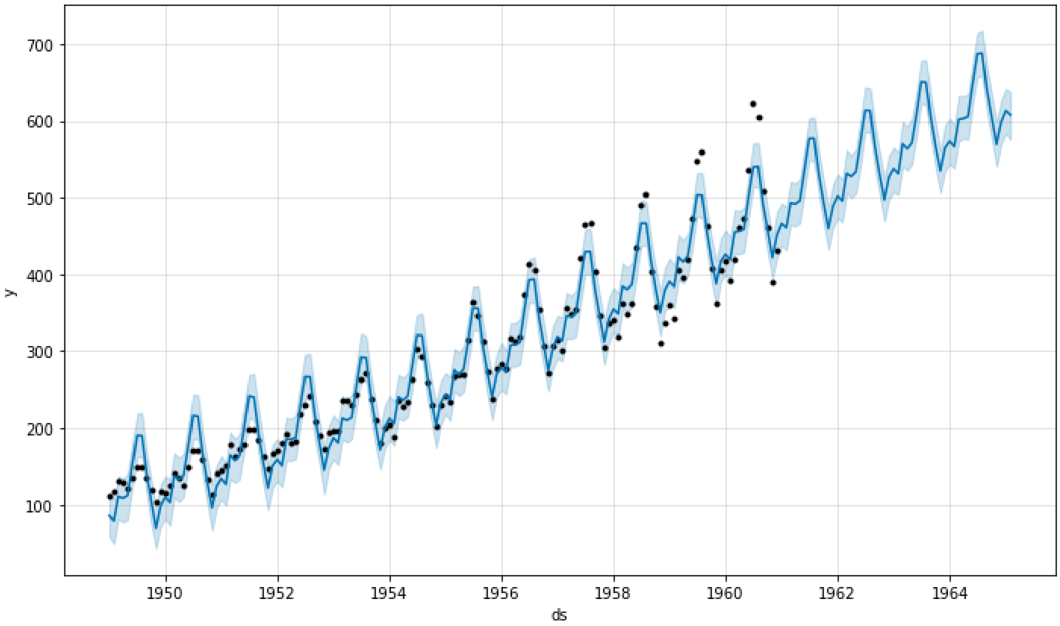
Prior probability shift: histogram of target variable Y

A prior probability shift can occur in cases where despite the input variables remain the same, our target variable changes. In our Probability of Default example, John may be facing the following situation: there could be some companies that were not really affected by the lockdown and have not suffered any revenue losses (e.g. pharmacies) but they deliberately chose not to repay their loan installments in order to save some money in view of worse days or because they know that the government may subsidize the loans of all companies (same X

3. Concept drift

A concept drift happens where the relations between the input and output variables change. So we are not anymore only focusing on X variables or only the Y variable but on the relations between them.

Definition 3. A concept drift is termed the situation where $P_{trn}(Y|X) \neq P_{tst}(Y|X)$.



Concept drift in time series problems: Airplane passengers. Source [2]

A concept drift may happen in situations where the data is trully temporal and thus depend heavily on time. For example, we might have built a machine learning model to predict daily number of flights in some airport. Due to economic bloom and other variables that have not been accounted on the model (latent variables), our target variable keeps changing over time (time series presents a *trend*). Another example is *selection bias*. This can happen when the train sample selected does not contain all the possible data distribution (it is a common caveat in questionaires and statistical surveys).

1. Covariate shift

To detect covariate shifts we are are going to use a simple but clever programmatic trick. We are actually going to deploy a machine learning solution for this goal. But this time, instead of trying to predict the target variable (whatever this is), we will build a classifier that will try to distinguish between the train and test sets. Seems confusing? Its really not, follow along and it will all make sense.

Approch description

Let's build on John's work. John initially trained a model on some data. We will call this data…the train set. He then deployed the model and every month he infers on a new dataset which shall be called…the test set. In order to check if the given test set is vastly different from the train set we are going to create a new dummy variable called 'is_train'. This variable will contain all ones (1) in the train set and all zeroes (0) in the test set. We will then use every indepedent variable, in turn, and on its own, to try to predict the new target variable 'is_train'. If an input variable is able to predict the new dummy variable, i.e. to separate between the train and test set, this means that this variable presents covariate shift between the train and test set and must be taken care of. In bullet-style we are going to follow the next steps in the next code chunk:

- Create new variable with ones in train set and zeroes in test set.
- Merge the two sets and shuffle randomly.
- Split in new train-test at 80%-20%
- For each single input variable:
    - Fit a simple classifier (e.g. Random Forests)
    - Predict 'is_train' variable
    - Calculate AUC
    - If AUC exceeds some threshold (e.g. 80%) feature displays data shift

a model interpretability technique (such as SHAP) to examine the impact of each variable in the prediction, therefore understanding which ones present covariate shift.

Code

The following chink of code does exactly what was described above: it creates a new variable to characterize the train and test sets and then tries to distinguish between them, using each variable on its own, iterativelly.

```
from sklearn.model_selection import train_test_split, cross_v
from sklearn.ensemble import RandomForestClassifier

# Create new y label to detect shift covariance
train['is_train'] = 1
test['is_train'] = 0

# Create a random index to extract random train and test samp
training = train.sample(7000, random_state=12)
testing = test.sample(7000, random_state=11)

## combining random samples
combi = training.append(testing)
y = combi['is_train']
combi.drop('is_train', axis=1, inplace=True)

## modelling
model = RandomForestClassifier(n_estimators=50, max_depth=5,
drop_list = []
score_list = []
temp = -1
for i in combi.columns[:50]:
score_list.append(np.mean(score))
print('checking feature no ', temp)
print(i, np.mean(score))
```

Running this code will output something like…

```
checking feature no 0 out of 2238

coupon_id 0.8002228979591837

checking feature no 1 out of 2238

customer_id 0.5554701428571429

checking feature no 2 out of 2238

age_range 0.5320016938775510

checking feature no 3 out of 2238

marital_status 0.5211866938775510
```

…depending on your specific features. We can then target specific

2. Prior probability shift

Approch description

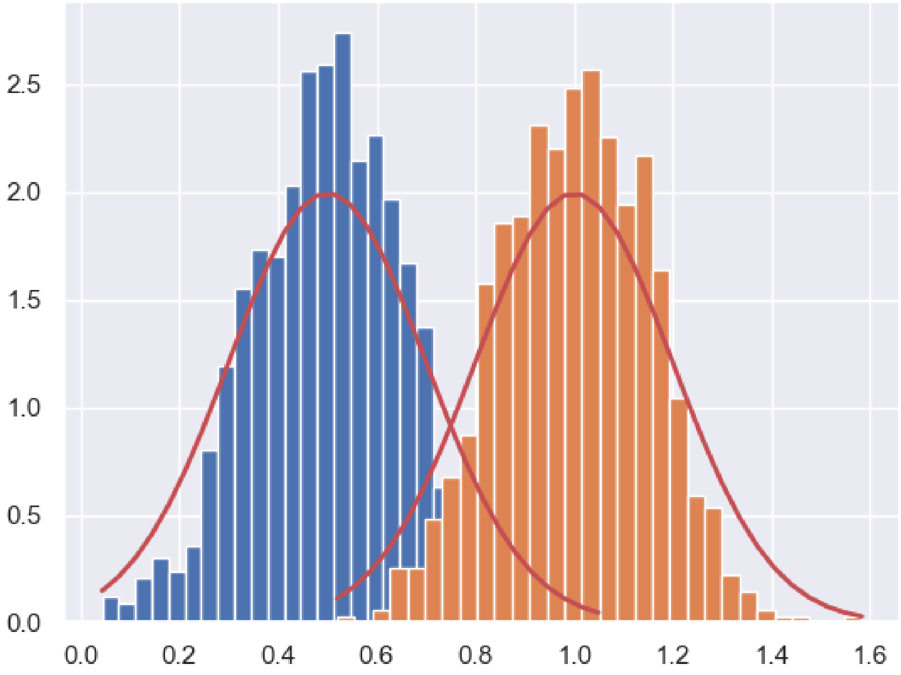
Code

You can run the following script to produce two normal distributio

```
mean = 0.5
std = 0.2
array = np.random.normal(0.5, 0.15, 1000)
count, bins, ignored = plt.hist(array, 30, normed=True)
plt.plot(bins, 1/(std * np.sqrt(2 * np.pi)) *
          np.exp( - (bins - mean)**2 / (2 * std**2) ),
          linewidth=2, color='r')
mean = 1
std = 0.2
array = np.random.normal(1, 0.15, 1000)
count, bins, ignored = plt.hist(array, 30, normed=True)
plt.plot(bins, 1/(std * np.sqrt(2 * np.pi)) *
          np.exp( - (bins - mean)**2 / (2 * std**2) ),
          linewidth=2, color='r')

plt.show()
```

Running this script will produce the following chart:



Target variable distribution change. Blue: train Y. Orange: T

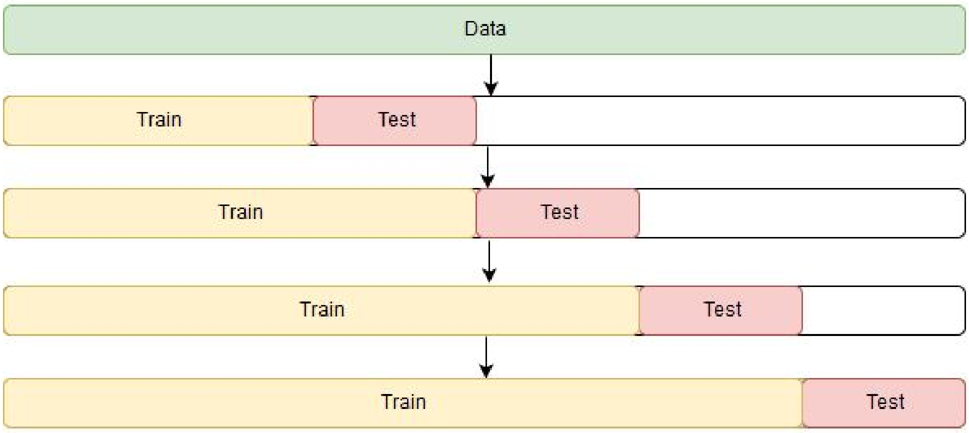
We can observe with a naked eye that the distributions of the

```
from scipy.stats import ttest_ind
ttest_ind(f1,f2)
```

The statistical test has a null hypothesis of mean simi

3 Concept drift

As already discussed, a concept drift generally arises w



Time-series split for temporal data illustration

I include a simple script below from sklearn to illustra

Code

```
from sklearn.model_selection import TimeSerie
from sklearn.linear_model import LogisticRegr
time_split = TimeSeriesSplit(n_splits=10)
logit = LogisticRegression(C=1, random_state=
cv_scores = cross_val_score(logit, X_train, y
```

A couple of last notes

Data shifts are very common in real world problems

Machine Learning models need often re-training to

Wrapping it up

In this post we described a very real but often ov

To avoid model degradation issues that arise from

P.S. The described 'John' story is entirely fictit

References

[1] https://towardsdatascience.com/understanding-d

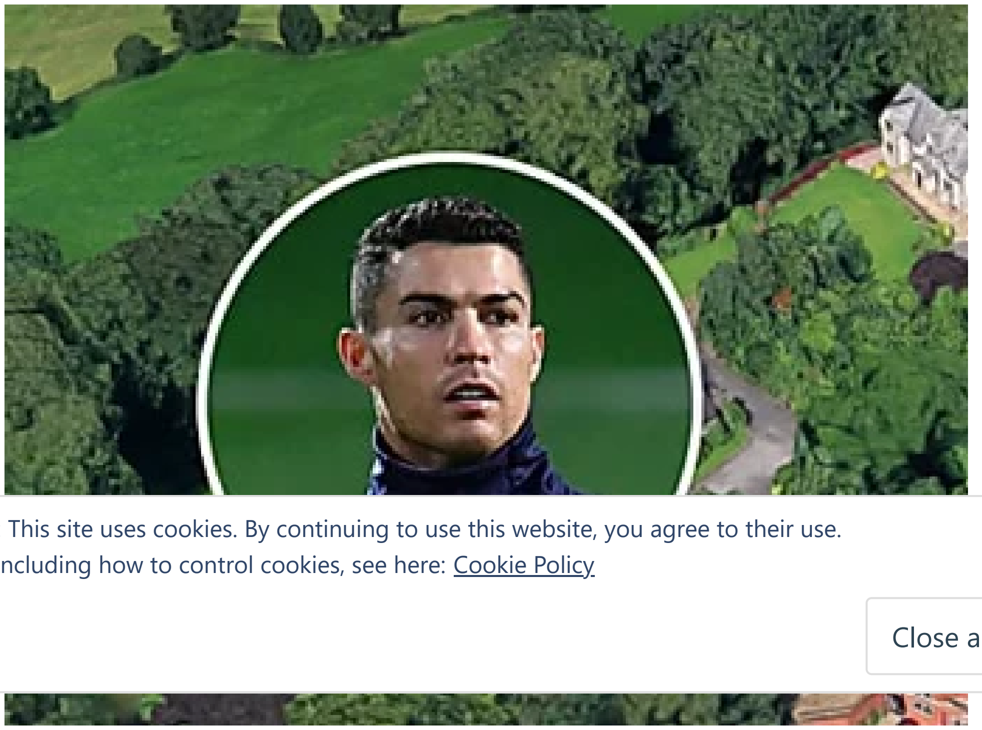
[2] https://facebook.github.io/prophet/docs/multip

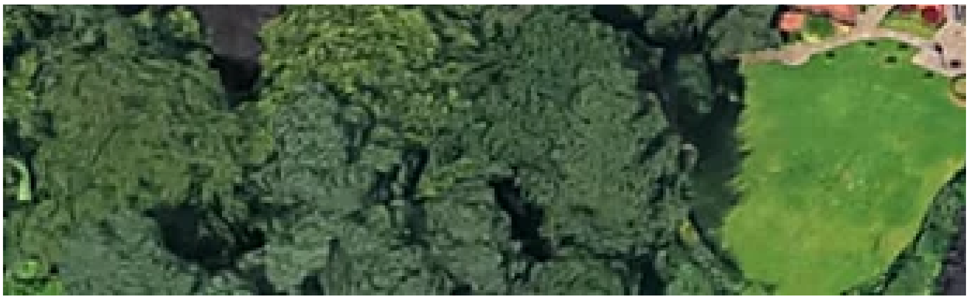
[3] https://docs.scipy.org/doc/scipy/reference/gen

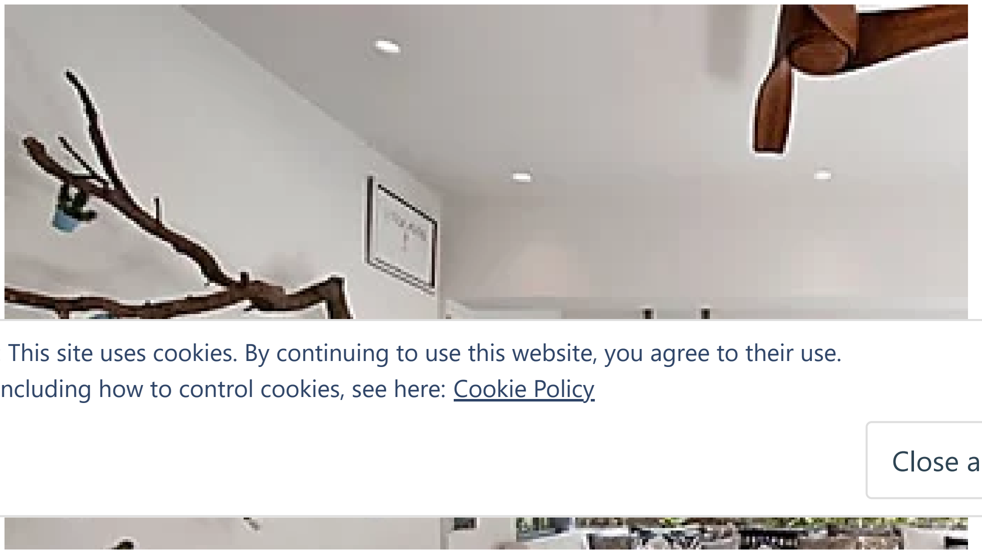
[4] https://mitpress.mit.edu/books/dataset-shift-m

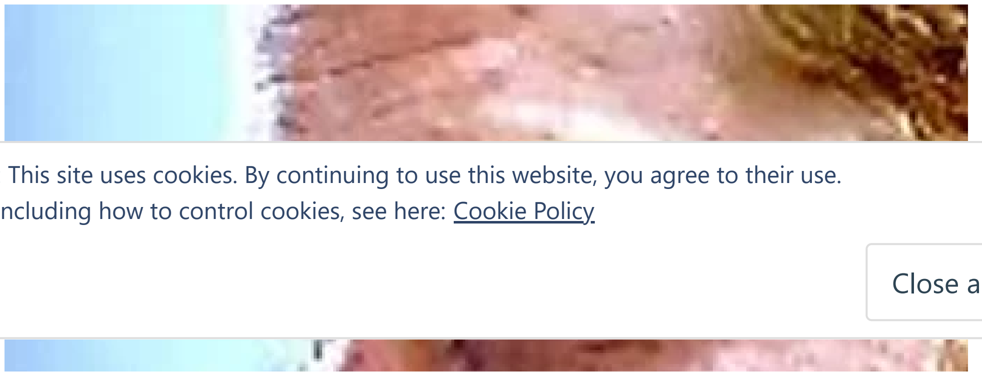
[6] https://scikit-learn.org/stable/modules/genera

[7] https://www.kaggle.com/kashnitsky/correct-time









Published by georsara1

View all posts by georsara1

Machine Learning, Uncategorized

data shift, datascience, Machine Learning, model degradation

Leave a Reply

Enter your comment here...

PREVIOUS

How to build a Machine Learning Football Predictions Website…

NEXT

The Confusion Matrix explained: a Business Perspective

BLOG STATS

39,424 hits