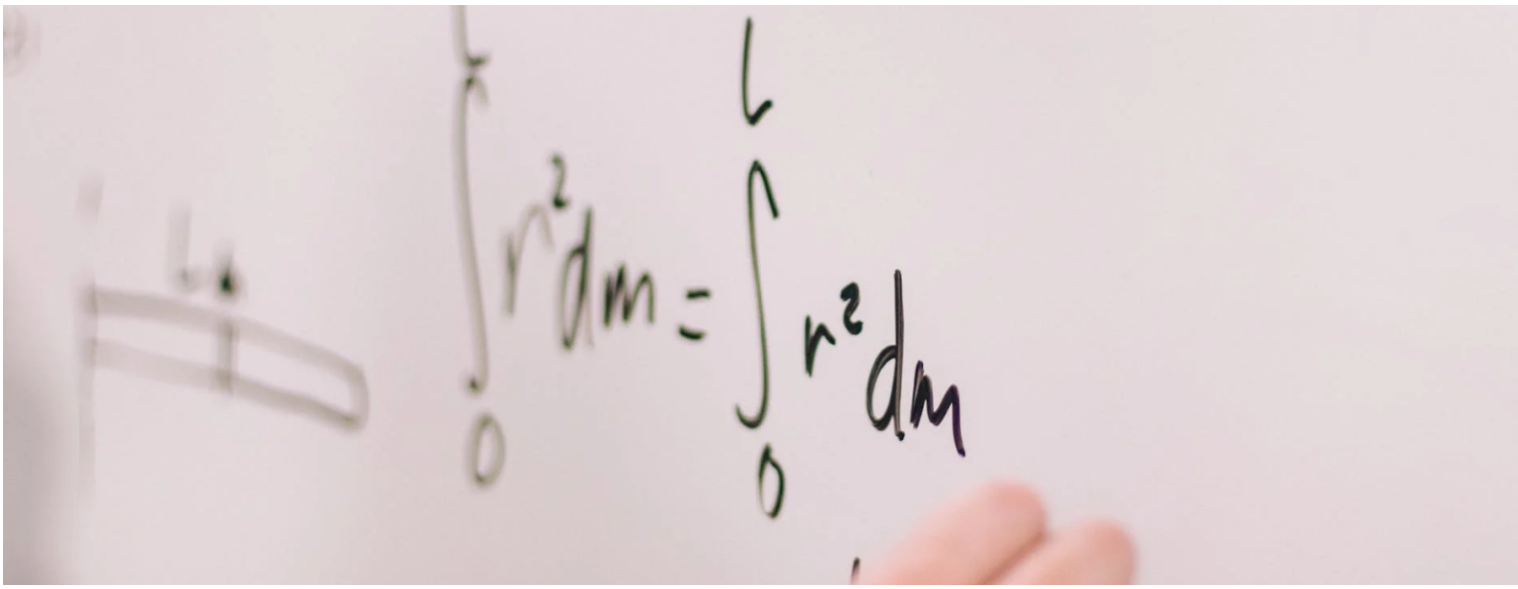
DEVELOPERS CORNER (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/DEVELOPERS_CORNER/)

Guide To AI Explainability 360: An Open Source Toolkit By IBM

BY AISHWARYA VERMA (HTTPS://ANALYTICSINDIAMAG.COM/AUTHOR/AISHWARYA-VERMAANALYTICSINDIAMAG-COM/)
29/01/2021



In our previous article, we detailed out the need for Trusted AI (https://analyticsindiamag.com/why-is-trusted-ai-so-important-and-how-to-build-it/) and discussed one of IBM Research Trusted AI toolkit called AIF360 (https://analyticsindiamag.com/guide-to-ai-fairness-360-an-open-source-toolkit-for-detection-and-mitigation-of-bias-in-ml-models/). I recommend you to read this article first for better understanding. In this article, we are going to discuss about AI Explainability 360 toolkit.

The growing interactions of the world with AI systems has pushed AI into all kinds of rigid domains(Agriculture, Law, Administration, etc) making predictions for society in all aspects(loan approval, cancer detection, etc). This has increased the onus of AI systems for being more reliable and accurate and providing a precise explanation for their decision-making process. These explanations allow users to get an insight into how a machine thinks which is important to gain trust

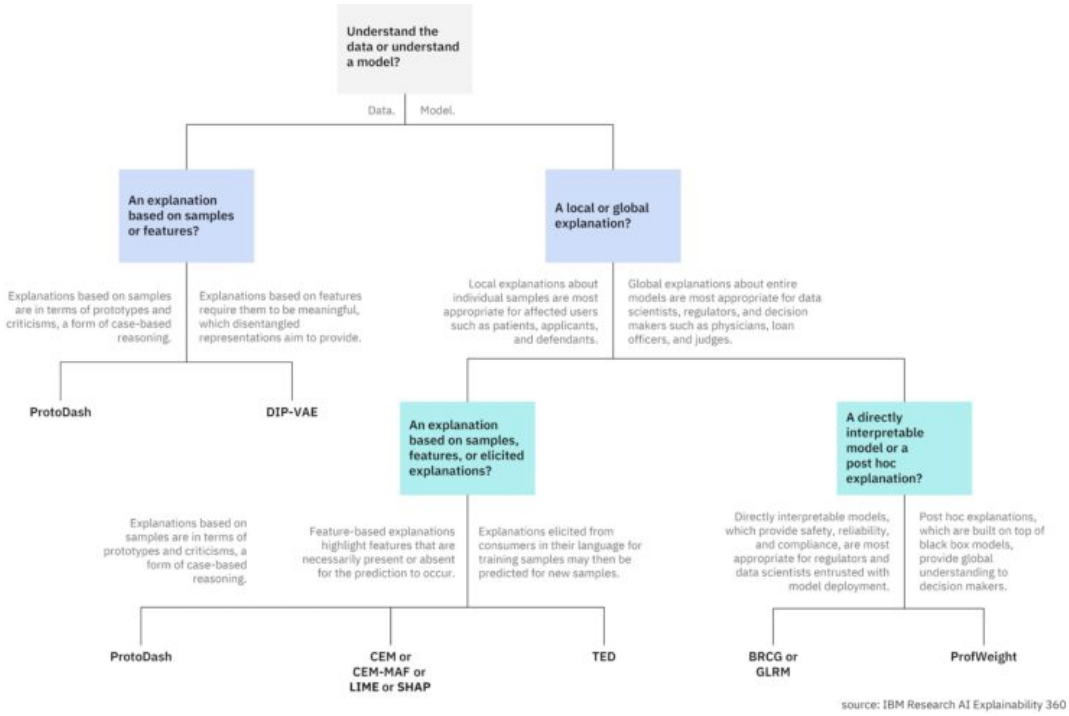
and confidence in AI systems. This has set off a growing community of researchers, developing **interpretable** or **explainable AI** (https://analyticsindiamag.com/how-to-obtain-explainability-in-ai-systems/) systems.

Though many algorithms and tools have come out, there is still a gap between what users want and what researchers are producing. The reason for this is the vague definition of an **explanation**. For instance, if a doctor tries to understand an AI system for cancer detection, he/she will look for a similar case and then conclude their result; If a person got rejected for loan application, he/she want to know the reasons behind it; However, if developers/ scientists deals with these case, they would want to understand where these AI models more or less confident as a means of improving its performance. Hence, we require some organizing principles and tools that can take account for different explanations, to solve this problem.

Keeping the above dilemma in mind, the research of IBM has come up with an AI Explainability 360(AIX 360 – One Explanation Does Not Fit All (https://arxiv.org/pdf/1909.03012.pdf)) toolkit. It is an open-source toolkit which takes account many possible explanations for consumers. The goal is to demonstrate how different explainability methods can be applied in real-world scenarios. It provides interpretability and explainability of datasets and machine learning models. The package for this toolkit is available in Python and includes a comprehensive set of algorithms that cover different dimensions of explanations along with explainability metrics. It gives the taxonomy of selecting the best possible explanation for the data as well as for models. Please refer to this link (https://aix360.mybluemix.net/resources#glossary) for the terminologies required in this session. There is no single approach to explainability. There are many ways to explain how machine learning makes predictions such as:

- data vs. model
- directly interpretable vs. post hoc explanation
- local vs. global
- static vs. interactive

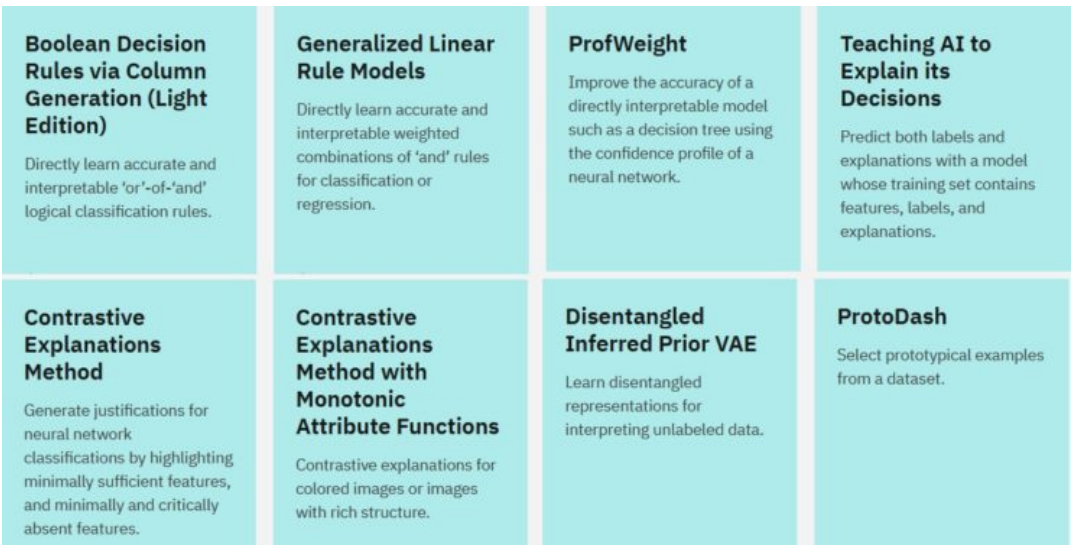
Given below is the required taxonomy for selecting the explanation algorithm, provided by AI Explainability 360.



Scroll To Top

There are eight state-of-the-art explainability algorithms that are added in AI Explainability 360 toolkit.

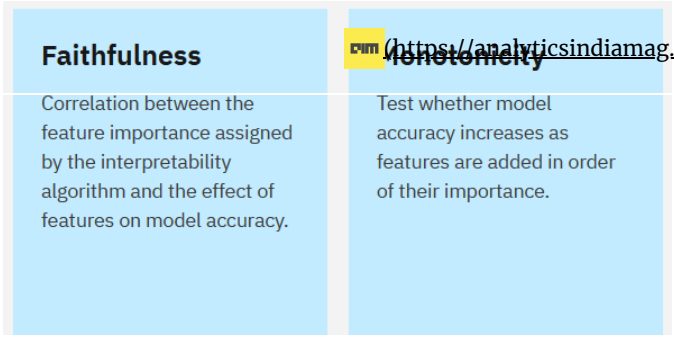
- ProtoDash (https://github.com/Trusted-AI/AIX360/blob/master/aix360/algorithms/protodash/PDASH.py)(Gurumoorthy et al., 2019 (https://arxiv.org/abs/1707.01212))
- Contrastive Explanations Method (https://github.com/IBM/AIX360/tree/master/aix360/algorithms/contrastive/CEM.py) (Dhurandhar et al., 2018 (https://papers.nips.cc/paper/7340-explanations-based-on-the-missing-towards-contrastive-explanations-with-pertinent-negatives))
- Contrastive Explanations Method with Monotonic Attribute Functions (https://github.com/Trusted-AI/AIX360/blob/master/aix360/algorithms/contrastive/CEM_MAF.py) (Luss et al., 2019 (https://arxiv.org/abs/1905.12698))
- LIME (Ribeiro et al. 2016 (https://arxiv.org/abs/1602.04938), Github (https://github.com/marcotcr/lime))
- SHAP (Lundberg, et al. 2017 (http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions), Github (https://github.com/slundberg/shap))
- Teaching AI to Explain its Decisions (https://github.com/Trusted-AI/AIX360/blob/master/aix360/algorithms/ted/TED_Cartesian.py) (Hind et al., 2019 (https://doi.org/10.1145/3306618.3314273))
- Boolean Decision Rules via Column Generation (https://github.com/Trusted-AI/AIX360/blob/master/aix360/algorithms/rbm/BRCG.py)(Light Edition) (Dash et al., 2018 (https://papers.nips.cc/paper/7716-boolean-decision-rules-via-column-generation))
- Generalized Linear Rule Models (https://github.com/Trusted-AI/AIX360/blob/master/aix360/algorithms/rbm/GLRM.py) (Wei et al., 2019 (http://proceedings.mlr.press/v97/wei19a.html))
- ProfWeight (https://github.com/Trusted-AI/AIX360/blob/master/aix360/algorithms/profwt/profwt.py) (Dhurandhar et al., 2018 (https://papers.nips.cc/paper/8231-improving-simple-models-with-confidence-profiles))



Source:https://aix360.mybluemix.net/

Supported explainability metrics are:

- Faithfulness (https://github.com/Trusted-AI/AIX360/blob/master/aix360/metrics/local_metrics.py)(Alvarez-Melis and Jaakkola, 2018 (https://papers.nips.cc/paper/8003-towards-robust-interpretability-with-self-explaining-neural-networks))
- Monotonicity (https://github.com/Trusted-AI/AIX360/blob/master/aix360/metrics/local_metrics.py)(Luss et al., 2019 (https://arxiv.org/abs/1905.12698))



Source:https://aix360.mybluemix.net/

Let's jump into the implementation part.

Requirements

1. Python (https://www.python.org/downloads/)>=3.6
2. Install the AIX 360 library through pip (https://pip.pypa.io/en/stable/)

```
!pip install aix360
```

Introduction to Dataset

For demo purposes, we are going to take a Credit Approval problem explaining different types of explanations required for data scientists, loan officers and bank customers. The dataset which we are going to use is the FICO Explainable Machine Learning Challenge (https://community.fico.com/s/explainable-machine-learning-challenge). The dataset can be downloaded by filling the google form here (https://community.fico.com/s/explainable-machine-learning-challenge?tabset-3158a=2)(for licensing) and an email will be sent to your registered email id containing the dataset and data dictionary. The details of the dataset can be found here (https://community.fico.com/s/explainable-machine-learning-challenge?tabset-3158a=2). The machine learning task for this dataset is to predict whether a customer has made payment within the time period or not. The target variable is *RiskPerformance*. The value "bad" indicates that the customer took more than 90 days for their due payments within 24 months of the account being open and the value "good" implies its opposite. The relationship between predictor variables and target variable is indicated by *Monotonicity Constraint* with the probability of bad equals to 1. If the value of this constraint is monotonically decreasing then as the value of variable increases, the probability of loan application being "bad" decreases.

Credit Loan Approval – Explanation for Data Scientists

1. Load the Heloc Dataset through AI Explainability 360 with *custom_preprocessing* equal to *nan_preprocessing*(convert special values to np.nan) which can directly be handled by AIX 360 algorithms(without replacing). Then split the data into train set and test set via sklearn (https://docs.google.com/document/d/1jh-l4kLns7jcrm704h-pIZP7OslEirzFjJgrMjkMn38/edit#%20Load%20FICO%20HELOC%20data%20with%20special%20values%20converted%20to%20np.nan%20from%20aix360.datasets.heloc_dataset%20import%20HELOCDataset,%20nan_preprocessing%20data%20=%20HELOCDataset(custom_preprocessing=nan_preprocessing).data()%20#%20Separate%20target%20variable%20y%20=%20data.pop('RiskPerformance')%20#%20Split%20data%20into%20training%20and%20test%20sets%20using%20fixed%20random%20seed%20from%20sklearn.model_selection%20import%20train_test_split%20dfTrain,%20dfTest,%20yTrain,%20yTest%20=%20train_test_split(data,%20y,%20random_state=0,%20stratify=y)%20dfTrain.head().transpose()).

```
# Load FICO HELOC data with special values converted to np.nan
from aix360.datasets.heloc_dataset import HELOCDataset, nan_preprocessing
data = HELOCDataset(custom_preprocessing=nan_preprocessing).data()
# Separate target variable
y = data.pop('RiskPerformance')
# Split data into training and test sets using fixed random seed
from sklearn.model_selection import train_test_split
dfTrain, dfTest, yTrain, yTest = train_test_split(data, y, random_state=0, stratify=y)
dfTrain.head().transpose()
```

2. We are going to use Boolean Decision Rules via Column Generation (https://github.com/Trusted-AI/AIX360/blob/master/aix360/algorithms/rbm/BRCG.py)(BRCG) and LogRR algorithms which require binary features. Hence, converting non-binarized features to binary form with the help of FeatureBinarizer class. The values are divided into 10 bins including all the continuous and negation values.

```
# Binarize data and also return standardized ordinal features
from aix360.algorithms.rbm import FeatureBinarizer
fb = FeatureBinarizer(negations=True, returnOrd=True)
##return standardized versions of the original unbinarized ordinal features, which are
used by LogRR but not BRCG
dfTrain, dfTrainStd = fb.fit_transform(dfTrain)
dfTest, dfTestStd = fb.transform(dfTest)
#Below is the result of binarizing the first 'ExternalRiskEstimate' feature.
dfTrain['ExternalRiskEstimate'].head()
```

3. BRCG is used to produce simple OR-of-Ands rule(disjunctive normal form, DNF) for binary classification. Here a DNF represents a whole rule set, where AND clause shows the individual value. Column Generation is used to generate promising results by searching for all possible rule sets. For this dataset, we have used CNF(conjunctive normal rule)

```
# Instantiate BRCG with small complexity penalty and large beam search width
from aix360.algorithms.rbm import BooleanRuleCG
# The model complexity parameters lambda0 and lambda1 penalize the number of clauses in
the
#rule and the number of conditions in each clause.
# We use the default values of 1e-3 for lambda0 and lambda1
#(decreasing them did not increase accuracy here) and leave other parameters at their
defaults as well.
# The model is then trained, evaluated, and printed.
br = BooleanRuleCG(lambda0=1e-3, lambda1=1e-3, CNF=True)

# Train, print, and evaluate model
br.fit(dfTrain, yTrain)
from sklearn.metrics import accuracy_score
print('Training accuracy:', accuracy_score(yTrain, br.predict(dfTrain)))
print('Test accuracy:', accuracy_score(yTest, br.predict(dfTest)))
print('Predict Y=0 if ANY of the following rules are satisfied, otherwise Y=1:')
print(br.explain()['rules'])
```

Here, ExternalRiskEstimate represents a consolidated version of risk markers(higher is better)and NumSatisfactoryTrades represents a number of satisfactory credit accounts. If the value of y is a “1” then the person defaulted on the loan. If the value is a “0” then the person paid back the loan. Point to be noted that with these two clauses including the same features, we are able to generate a pretty decent accuracy of 69.65%.

4. Next we fit and evaluate the LogRR model, which improves the accuracy by making the model a bit complex. It fits a logistic regression algorithm using rule-based features.

Scroll To Top

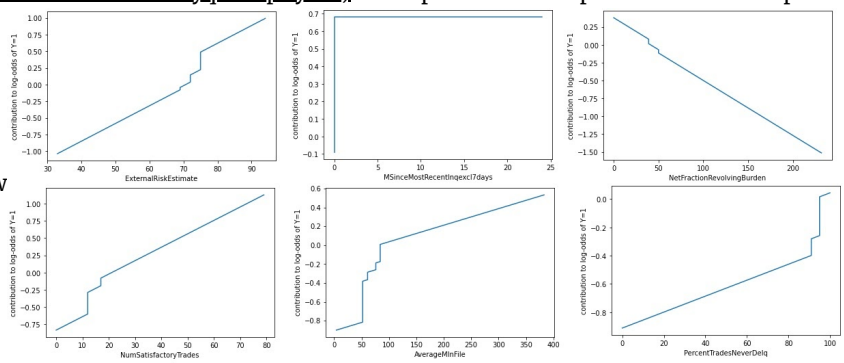
```
# Instantiate LRR with good complexity penalty and numerical features
from aix360.algorithms.rbm import LogisticRuleRegression
# Here we are also including unbinarized ordinal features (useOrd=True) in addition to
rules.
# Similar to BRCG, the complexity parameters lambda0, lambda1 penalize the number of
rules included in the model and the number of conditions in each rule.
# The values for lambda0, lambda1 below strike a good balance between accuracy and model
complexity.

lrr = LogisticRuleRegression(lambda0=0.005, lambda1=0.001, useOrd=True)

# Train, print, and evaluate model
lrr.fit(dfTrain, yTrain, dfTrainStd)
print('Training accuracy:', accuracy_score(yTrain, lrr.predict(dfTrain, dfTrainStd)))
print('Test accuracy:', accuracy_score(yTest, lrr.predict(dfTest, dfTestStd)))
print('Probability of Y=1 is predicted as logistic(z) = 1 / (1 + exp(-z))')
print('where z is a linear combination of the following rules/numerical features:')
lrr.explain()
```

The test accuracy of the LogRR model is slightly better than the BRCG. The model created all the rules with single conditions only, no interactions are made between the features. Hence, it is a kind of generalized additive model (GAM) (https://en.wikipedia.org/wiki/Generalized_additive_model). Next, we will see the plotting of univariate functions that make it as GAM.

5. With the help of the visualize() method of LogRR model, we will plot all the first degree rules. Out of 36 features, 14(excluding intercept) features are single conditioned. Check here (https://colab.research.google.com/drive/12IwxXr1e1V9IavWOWmUcRUGcyYscA6PF?authuser=1#scrollTo=8yqB0WpTyzro)all the plots and its explanation. An example of it is shown below



You can check the full demo for this, here (https://colab.research.google.com/drive/12IwxXr1e1V9IavWOWmUcRUGcyYscA6PF?usp=sharing).

Credit Loan Approval – Explanation for Loan Officers

In this section, we will take an algorithmic approach to give explanations for the AI model that will help Loan officers understand the black box ML. For this, we will try to find similar user profiles for an applicant obtained by Protodash Algorithm (https://arxiv.org/abs/1707.01212). This algorithm selects the application from the training dataset that is similar to the applicant, we want to explain and this similarity is found in different ways(by matching the distribution), unlike nearest neighbour techniques. Hence, Protodash gives a wholesome view to answer why the particular decision has been made.

We will see two cases: One, where the application has been approved. Second, where the application has been rejected. In each case, the top five prototypes from the training data along with the similarity factor will be explained.

1. Import the required libraries and framework

```
#Import necessary libraries, frameworks and a
import pandas as pd
import numpy as np
import tensorflow as tf
from keras.models import Sequential, Model, load_model, model_from_json
from keras.layers import Dense
import matplotlib.pyplot as plt
from IPython.core.display import display, HTML

from aix360.algorithms.contrastive import CEMExplainer, KerasClassifier
from aix360.algorithms.protodash import ProtodashExplainer
from aix360.datasets.heloc_dataset import HELOCDataset
```

2. Load the dataset and show some sample applicants.

```
# Clean data and split dataset into train/test
(Data, x_train, x_test, y_train_b, y_test_b) = heloc.split()

#Normalize the dataset
Z = np.vstack((x_train, x_test))
Zmax = np.max(Z, axis=0)
Zmin = np.min(Z, axis=0)

#normalize an array of samples to range [-0.5, 0.5]
def normalize(V):
    VN = (V - Zmin)/(Zmax - Zmin)
    VN = VN - 0.5
    return(VN)

# rescale a sample to recover original values for normalized values.
def rescale(X):
    return(np.multiply ( X + 0.5, (Zmax - Zmin) ) + Zmin)

N = normalize(Z)
xn_train = N[0:x_train.shape[0], :]
xn_test  = N[x_train.shape[0]:, :]
```

4. Define the Neural Network and train the dataset. The architecture of neural network can be defined as:

```
#this is the architecture of a 2-layer neural network classifier whose predictions we
will try to interpret.
# nn with no softmax
def nn_small():
    model = Sequential()
    model.add(Dense(10, input_dim=23, kernel_initializer='normal', activation='relu'))
    model.add(Dense(2, kernel_initializer='normal'))
    return model
```

Now. train this neural network.

Scroll To Top

```
# Set random seeds for repeatability
np.random.seed(1)
tf.set_random_seed(2)

class_names = ['Bad', 'Good']

# loss function
def fn(correct, predicted):
    return tf.nn.softmax_cross_entropy_with_logits(labels=correct, logits=predicted)

# compile and print model summary
nn = nn_small()
nn.compile(loss=fn, optimizer='adam', metrics=['accuracy'])
nn.summary()

# train model or load a trained model
TRAIN_MODEL = True

if (TRAIN_MODEL):
    nn.fit(xn_train, y_train_b, batch_size=128, epochs=500, verbose=1, shuffle=False)
    nn.save_weights("heloc_nnsmall.h5")
else:
    nn.load_weights("heloc_nnsmall.h5")

# evaluate model accuracy
score = nn.evaluate(xn_train, y_train_b, verbose=0) #Compute training set accuracy
#print('Train loss:', score[0])
print('Train accuracy:', score[1])

score = nn.evaluate(xn_test, y_test_b, verbose=0) #Compute test set accuracy
#print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

5. Obtaining similar applicants whose application got approved.

```
#normalize the data and chose a particular applicant, whose profile is displayed below
p_train = nn.predict_classes(xn_train) # Use trained neural network to predict train
points
p_train = p_train.reshape((p_train.shape[0],1))

z_train = np.hstack((xn_train, p_train)) # Store (normalized) instances that were
predicted as Good
z_train_good = z_train[z_train[:,-1]==1, :]

zun_train = np.hstack((x_train, p_train)) # Store (unnormalized) instances that were
predicted as Good
zun_train_good = zun_train[zun_train[:,-1]==1, :]
```

Now, we will choose applicant 8 whose application got approved. The explanation for these will be provided by Protodash Algorithm.

```
##Let us now consider applicant 8 whose loan was approved!
#Note that this applicant was also considered for the contrastive explainer, however,
#we now justify the approved status in a different manner using prototypical examples,
#which is arguably a better explanation for a bank employee.

idx = 8

X = xn_test[idx].reshape((1,) + xn_test[idx].shape)

print("Chosen Sample:", idx)
print("Prediction made by the model:", class_names[np.argmax(nn.predict_proba(X))])
print("Prediction probabilities:", nn.predict_proba(X))
print("")

# attach the prediction made by the model to X
X = np.hstack((X, nn.predict_classes(X).reshape((1,1))))

Xun = x_test[idx].reshape((1,) + x_test[idx].shape)
dfx = pd.DataFrame.from_records(Xun.astype('double')) # Create dataframe with original
feature values
dfx[23] = class_names[int(X[0, -1])]
dfx.columns = df.columns
dfx.transpose()
```

Now, fitting the protodash algorithm to find the similar applicants predicted as “Good”.

```
#Find similar applicants predicted as "good" using the protodash explainer.
explainer = ProtodashExplainer()
(W, S, setValues) = explainer.explain(X, z_train_good, m=5) # Return weights W,
Prototypes S and objective function values
```

Display similar applicants along with similarity factors labelled as “weights”.

```
## Display similar applicant user-profiles and the extent to which they are similar to
the chosen applicant
##as indicated by the last row in the table below labelled as "Weight"

dfs = pd.DataFrame.from_records(zun_train_good[S, 0:-1].astype('double'))
RP=[]
for i in range(S.shape[0]):
    RP.append(class_names[int(z_train_good[S[i], -1])]) # Append class names
dfs[23] = RP
dfs.columns = df.columns
dfs["Weight"] = np.around(W, 5)/np.sum(np.around(W, 5)) # Calculate normalized importance
weights
dfs.transpose()
```

Now, compute and display the similarity of features of prototypical users to the chosen applicant. The closer the value to 1, the more similar the features of prototypical users to the chosen applicant.

Scroll To Top

```
# ##Compute how similar a feature of a prototypical user is to the chosen applicant.
# The more similar the feature of prototypical user is to the applicant, the closer its
weight is to 1.
# We can see below that several features for prototypes are quite similar to the chosen
applicant.
# A human friendly explanation is provided thereafter.
z = z_train_good[S, 0:-1] # Store chosen prototypes
eps = 1e-10 # Small constant defined to eliminate divide-by-zero errors
fwt = np.zeros(z.shape)
for i in range (z.shape[0]):
    for j in range(z.shape[1]):
        fwt[i, j] = np.exp(-1 * abs(X[0, j] - z[i,j])/(np.std(z[:, j])+eps)) # Compute
feature similarity in [0,1]

# move wts to a dataframe to display
dfw = pd.DataFrame.from_records(np.around(fwt.astype('double'), 2))
dfw.columns = df.columns[:-1]
dfw.transpose()
```

The output of above code is displayed below:

	0	1	2	3	4		0	1	2	3	4
ExternalRiskEstimate	0.59	0.29	0.42	0.84	0.21	NumTradesOpeninLast12M	1.00	1.00	0.40	0.40	0.06
MSinceOldestTradeOpen	0.76	0.62	0.76	0.09	0.79	PercentInstallTrades	1.00	0.05	0.54	0.37	0.33
MSinceMostRecentTradeOpen	1.00	0.09	0.83	0.89	0.87	MSinceMostRecentInqexcl7days	0.08	1.00	1.00	1.00	1.00
AverageMInFile	0.79	0.09	0.90	1.00	0.82	NumInqLast6M	0.21	1.00	0.21	0.21	0.04
NumSatisfactoryTrades	0.95	0.39	0.74	0.39	0.15	NumInqLast6Mexcl7days	0.26	1.00	0.26	1.00	0.07
NumTrades60Ever2DerogPubRec	1.00	1.00	0.08	1.00	1.00	NetFractionRevolvingBurden	0.96	0.88	0.96	0.92	0.09
NumTrades90Ever2DerogPubRec	1.00	1.00	0.08	1.00	1.00	NetFractionInstallBurden	1.00	1.00	1.00	1.00	0.08
PercentTradesNeverDelq	1.00	0.15	0.81	0.15	0.15	NumRevolvingTradesWBalance	1.00	0.28	0.38	0.73	0.20
MSinceMostRecentDelq	1.00	0.36	0.22	0.36	0.36	NumInstallTradesWBalance	1.00	0.13	1.00	0.13	1.00
MaxDelq2PublicRecLast12M	1.00	0.13	1.00	0.13	1.00	NumBank2NatlTradesWHighUtilization	0.69	0.69	0.69	1.00	0.11
MaxDelqEver	1.00	0.41	0.17	0.41	0.64	PercentTradesWBalance	0.67	0.12	0.36	0.38	0.57
NumTotalTrades	0.80	0.23	0.86	0.26	0.35						

The above table displays the five closed user profiles for the chosen applicant. We can conclude from the table that, out of 5 user profiles, the user 0 is the most similar applicant to the given user(12 out of 23 features are exactly similar i.e., weight is equal to 1). In all, this table strongly proves that the chosen applicant is no defaulter. Hence, giving more trust to Loan Officers to mark the application as "approved".

6. Similarly, we can give explanations for a rejected applicant. The code for this is similar to the approved applicant, you can check the code snippet here (https://colab.research.google.com/drive/1jp0W1uzyVXYLU7q1D36RajSS54qs7pU8?authuser=1#scrollTo=rpAENWBtS99M).

You can check the full demo, here (https://colab.research.google.com/drive/1jp0W1uzyVXYLU7q1D36RajSS54qs7pU8?usp=sharing).

Credit Loan Approval – Explanation for Customers

In this part of the demo, we are going to compute explanations for end-users(customers) of AI systems that can make users understand why their application is approved or rejected and what are the required changes to change the decision of AI model from reject to approve. Apart from that, organizations(banks, financial institutes, etc.) also want to understand the approach of the AI model in approving or rejecting the application. We will try to find out these contrastive explanations with help of **Explanations based on the Missing: Towards Contrastive Explanations with Pertinent Negatives** (https://arxiv.org/abs/1802.07623) algorithm. This algorithm consists of two parts:

a) Pertinent Negatives (PNs): It identifies a minimal set of features which if altered would change the classification of the original input. For example, in this case if a person's credit score is increased their loan application status may change from reject to accept.

Scroll To Top

b) Pertinent Positives (PPs) : It identifies a minimal set of features and their values that are sufficient to yield the original input's classification. For example, an individual's loan may still be accepted if the salary was 50K as opposed to 100K.

For more details of this algorithm, please refer to the above link. The initial 4 steps for generating contrastive explanations are the same as we discussed in this (https://docs.google.com/document/d/1Vew78sfL4HhdSbOndUnQer9PulgB27_AVztjOCn5RmM/edit#bookmark=kix.e59aglcx4009)section.

1. Import the required libraries and framework(same as above).
2. Load the dataset and show some sample applicants(same as above).
3. Preprocess and normalize the Heloc dataset for training(same as above).
4. Define the Neural Network and train the dataset(same as above).
5. Now, compute the contrastive explanation for a few applicants. For the first part, we will select the applicants whose application was rejected and will try to find out the minimal change required to change the status to approved(by calculating Pertinent Negatives). For the second part. We will select the applicants whose application was approved and will try to find the minimal values of features that will yield the same result(i.e., calculating Positive Pertinent).

First, the computer PN's CEM explainer will calculate the similar user profiles(for chosen applicants) whose result is not the same as the chosen applicant. This explainer will change the minimal set of features to a minimal amount and try to learn on what changes, the chosen application got selected.

```
# In order to compute pertinent negatives, the CEM explainer computes a user profile that
is close to the original applicant but
# for whom the decision of HELOC application is different. The explainer alters a minimal
set of features by a minimal (positive) amount.
# This will help the user whose loan application was initially rejected say, to ascertain
how to get it accepted.

# Some interesting user samples to try: 2344 449 1168 1272
idx = 1272

X = xn_test[idx].reshape((1,) + xn_test[idx].shape)
print("Computing PN for Sample:", idx)
print("Prediction made by the model:", nn.predict_proba(X))
print("Prediction probabilities:", class_names[np.argmax(nn.predict_proba(X))])
print("")

mymodel = KerasClassifier(nn)
explainer = CEMExplainer(mymodel)

arg_mode = 'PN' # Find pertinent negatives
arg_max_iter = 1000 # Maximum number of iterations to search for the optimal PN for given
parameter settings
arg_init_const = 10.0 # Initial coefficient value for main loss term that encourages
class change
arg_b = 9 # No. of updates to the coefficient of the main loss term
arg_kappa = 0.2 # Minimum confidence gap between the PNs (changed) class probability and
original class' probability
arg_beta = 1e-1 # Controls sparsity of the solution (L1 loss)
arg_gamma = 100 # Controls how much to adhere to a (optionally trained) auto-encoder
my_AE_model = None # Pointer to an auto-encoder
arg_alpha = 0.01 # Penalizes L2 norm of the solution
arg_threshold = 1. # Automatically turn off features <= arg_threshold if arg_threshold <
1
arg_offset = 0.5 # the model assumes classifier trained on data normalized
                 # in [-arg_offset, arg_offset] range, where arg_offset is 0 or 0.5
# Find PN for applicant 1272
(adv_pn, delta_pn, info_pn) = explainer.explain_instance(X, arg_mode, my_AE_model,
arg_kappa, arg_b,
                                                         arg_max_iter, arg_init_const,
arg_beta, arg_gamma,
                                                            arg_alpha, arg_threshold,
arg_offset)
```

Now, we will examine an applicant whose application got rejected, with the help of PN's. We will also generate the importance of each feature to convert the result from negative to positive.

Scroll To Top

```
 # Let us start by examining one particular loan application that was denied for
applicant 1272.
# We showcase below how the decision could have been different through minimal changes to
the profile conveyed by the pertinent negative.
#  We also indicate the importance of different features to produce the change in the
application status.
# The column delta in the table below indicates the necessary deviations for each of the
features to produce this change.
# A human friendly explanation is then provided based on these deviations following the
feature importance plot.

#copying the negative peritnent value to a new variable
Xpn = adv_pn
classes = [ class_names[np.argmax(nn.predict_proba(X))],
class_names[np.argmax(nn.predict_proba(Xpn))], 'NIL' ]

print("Sample:", idx)
#Making prediction based on the original features
print("prediction(X)", nn.predict_proba(X), class_names[np.argmax(nn.predict_proba(X))])
#Making predictions based on the altered features
print("prediction(Xpn)", nn.predict_proba(Xpn),
class_names[np.argmax(nn.predict_proba(Xpn))] )


X_re = rescale(X) # Convert values back to original scale from normalized
Xpn_re = rescale(Xpn)
Xpn_re = np.around(Xpn_re.astype(np.double), 2)

delta_re = Xpn_re - X_re
delta_re = np.around(delta_re.astype(np.double), 2)
delta_re[np.absolute(delta_re) < 1e-4] = 0

X3 = np.vstack((X_re, Xpn_re, delta_re))

dfre = pd.DataFrame.from_records(X3) # Create dataframe to display original point, PN and
difference (delta)
dfre[23] = classes

dfre.columns = df.columns
dfre.rename(index={0:'X',1:'X_PN', 2:'(X_PN - X)'}, inplace=True)
dfret = dfre.transpose()


def highlight_ce(s, col, ncols):
    if (type(s[col]) != str):
        if (s[col] > 0):
            return(['background-color: yellow']*ncols)
    return(['background-color: white']*ncols)

dfret.style.apply(highlight_ce, col='(X_PN - X)', ncols=3, axis=1)
```

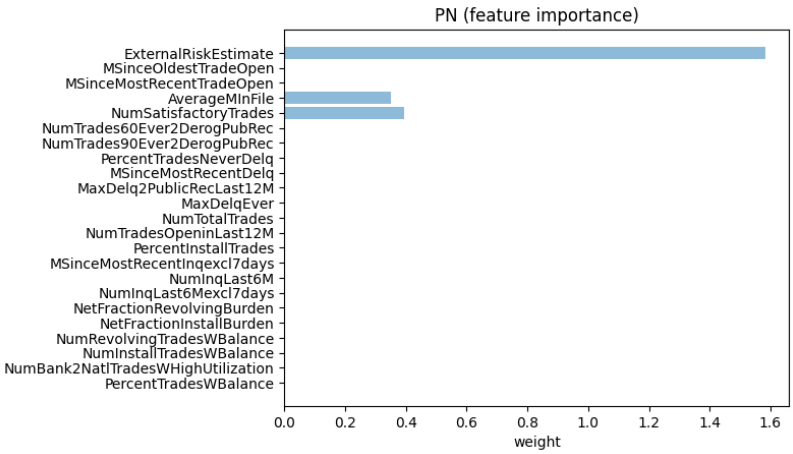
The output of the above code is the table indicating all the altered values in feature to generate the required change in result. The last column (X__PN-X) details out the necessary deviation required for the required change. The value “0” indicates that no change is required.

	X	X_PN	(X_PN - X)
ExternalRiskEstimate	65.000000	81.000000	16.000000
MSinceOldestTradeOpen	256.000000	256.000000	0.000000
MSinceMostRecentTradeOpen	15.000000	15.000000	0.000000
AverageMInFile	52.000000	63.990000	11.990000
NumSatisfactoryTrades	17.000000	21.460000	4.460000
NumTrades60Ever2DerogPubRec	0.000000	0.000000	0.000000
NumTrades90Ever2DerogPubRec	0.000000	0.000000	0.000000
PercentTradesNeverDelq	100.000000	100.000000	0.000000
MSinceMostRecentDelq	0.000000	0.000000	0.000000
MaxDelq2PublicRecLast12M	7.000000	7.000000	0.000000
MaxDelqEver	8.000000	8.000000	0.000000
NumTotalTrades	19.000000	19.000000	0.000000

	X	X_PN	(X_PN - X)
NumTradesOpeninLast12M	0.000000	0.000000	0.000000
PercentInstallTrades	29.000000	29.000000	0.000000
MSinceMostRecentInqexcl7days	2.000000	2.000000	0.000000
NumInqLast6M	5.000000	5.000000	0.000000
NumInqLast6Mexcl7days	5.000000	5.000000	0.000000
NetFractionRevolvingBurden	57.000000	57.000000	0.000000
NetFractionInstallBurden	79.000000	79.000000	0.000000
NumRevolvingTradesWBalance	2.000000	2.000000	0.000000
NumInstallTradesWBalance	4.000000	4.000000	0.000000
NumBank2NatlTradesWHighUtilization	2.000000	2.000000	0.000000
PercentTradesWBalance	60.000000	60.000000	0.000000
RiskPerformance	Bad	Good	NIL

Scroll To Top

Now, to translate these changes(from above table) in an easier way(for the customer), we will plot the above table and try to find out the importance of each PN feature. The code snippet is available here (https://colab.research.google.com/drive/1jKtJZoFt5ZcdzPb--EmygvTEgp5eYn2a?authuser=1#scrollTo=DAm4vGptpn7g&line=4&uniqifier=1).



This indicates that the applicant 1272's loan application would have been accepted if:

- ExternalRiskEstimate(credit score risk) increased from 65 to 81.
- AverageMInFile increased from 52 months to 66 months.
- And lastly, NumSatisfactoryTrades increased from 17 to 21.

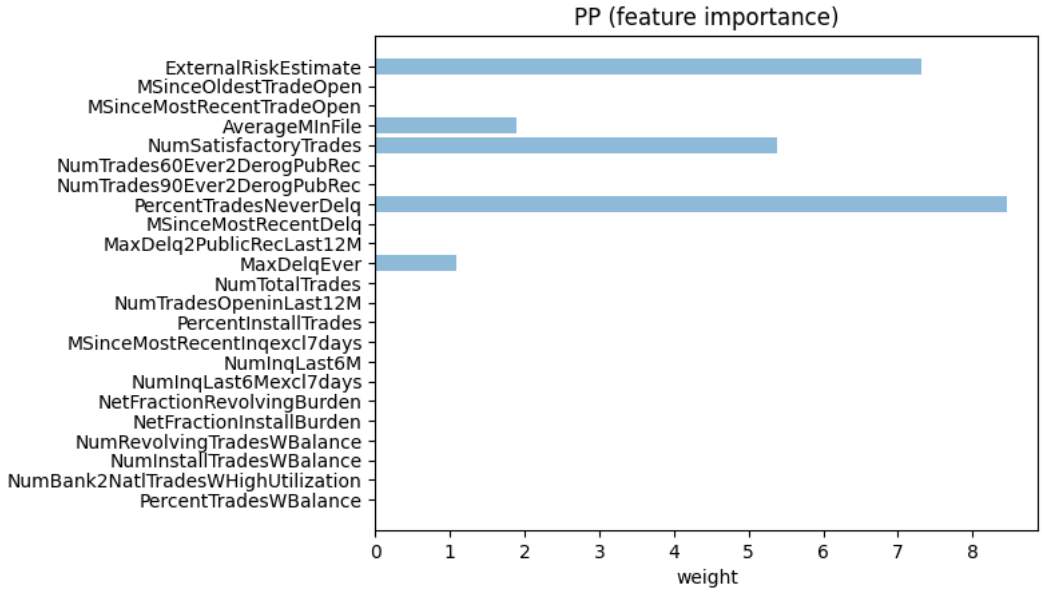
The above changes reflect that the chance of application being approved may increase. It does not guarantee the required change in result.

6. Similarly, we can calculate the Positive pertinents to know about the minimal sufficient values of each feature, keeping the result the same in this case. It will be opposite to what we did for Negative Pertinents. The code snippet for this is available here (https://colab.research.google.com/drive/1jKtJZoFt5ZcdzPb--EmygvTEgp5eYn2a?authuser=1#scrollTo=PCCvNaiYy4Ad)and the table generated is shown below.

	X	X_PP
ExternalRiskEstimate	74.000000	74.000000
MSinceOldestTradeOpen	181.000000	0.000000
MSinceMostRecentTradeOpen	1.000000	0.000000
AverageMInFile	65.000000	65.000000
NumSatisfactoryTrades	61.000000	61.000000
NumTrades60Ever2DerogPubRec	0.000000	0.000000
NumTrades90Ever2DerogPubRec	0.000000	0.000000
PercentTradesNeverDelq	100.000000	100.000000
MSinceMostRecentDelq	0.000000	0.000000
MaxDelq2PublicRecLast12M	6.000000	0.000000
MaxDelqEver	7.000000	2.000000
NumTotalTrades	65.000000	0.000000

	X	X_PP
NumTradesOpeninLast12M	5.000000	0.000000
PercentInstallTrades	35.000000	0.000000
MSinceMostRecentInqexcl7days	0.000000	0.000000
NumInqLast6M	0.000000	0.000000
NumInqLast6Mexcl7days	0.000000	0.000000
NetFractionRevolvingBurden	12.000000	0.000000
NetFractionInstallBurden	80.000000	0.000000
NumRevolvingTradesWBalance	9.000000	0.000000
NumInstallTradesWBalance	6.000000	0.000000
NumBank2NatlTradesWHighUtilization	2.000000	0.000000
PercentTradesWBalance	58.000000	0.000000
RiskPerformance	Good	Good

In the above table, the yellow marks in X_PP represent the minimal values of each feature that won't affect the result. Here, value "0" represents that those features are not important. The whole table can be summarised in the graph below.



This graph indicates that the 9's loan application would remain accepted if there is:

- No change in ExternalRiskEstimate, AverageMInFile, NumSatisfactoryTrades, PercentTradesNeverDelq
- And the value of MaxDelqEver is at least 2.

You can check the full demo of this, here (https://colab.research.google.com/drive/1jKtJZoFt5ZcdzPb--EmygvTEgp5eYn2a?usp=sharing).

Endnotes

- Colab Notebook AI Explainability 360 – For data scientist explanation (https://colab.research.google.com/drive/12IwxXr1e1V9IavWOWmUcRUGcyYscA6PF?usp=sharing)
- Colab Notebook AI Explainability 360 – For Loan officer explanation (https://colab.research.google.com/drive/1jp0W1uzyVXYLU7q1D36RajSS54qs7pU8?usp=sharing)
- Colab Notebook AI Explainability 360 – For Customer explanation (https://colab.research.google.com/drive/1jKtJZoFt5ZcdzPb--EmygvTEgp5eYn2a?usp=sharing)

Other examples of using AIX 360 are:

- Medical Expenditure (https://nbviewer.jupyter.org/github/IBM/AIX360/blob/master/examples/tutorials/MEPS.ipynb)
- Dermoscopy (https://nbviewer.jupyter.org/github/IBM/AIX360/blob/master/examples/tutorials/dermoscopy.ipynb)
- Health and Nutrition Survey (https://nbviewer.jupyter.org/github/IBM/AIX360/blob/master/examples/tutorials/CDC.ipynb)
- Proactive Retention (https://nbviewer.jupyter.org/github/IBM/AIX360/blob/master/examples/tutorials/retention.ipynb)

You can check other toolkits developed by IBM Research Trusted AI here:

- AI Fairness 360 (https://aif360.mybluemix.net/?_ga=2.5323575.507816120.1611548442-674595796.1611290149)(article (https://analyticsindiamag.com/guide-to-ai-fairness-360-an-open-source-toolkit-for-detection-and-mitigation-of-bias-in-ml-models/))
- Adversarial Robustness 360 (https://www.ibm.com/blogs/research/2019/09/adversarial-robustness-360-toolbox-v1-0/?_ga=2.5323575.507816120.1611548442-674595796.1611290149&cm_mc_uid=31592805874101611290148 9&cm_mc_sid_50200000=68499481611689680555)(article (https://analyticsindiamag.com/adversarial-robustness-toolbox-art/))
- AI FactSheets (https://aifs360.mybluemix.net/?_ga=2.5699767.507816120.1611548442-674595796.1611290149)(article (https://analyticsindiamag.com/ibm-commercialises-its-ai-factsheets-could-it-become-an-industry-standard/))

Resources and tutorial used above:

- Research Paper (https://arxiv.org/pdf/1909.03012.pdf)
- Github (https://github.com/Trusted-AI/AIX360)
- Website (https://aix360.mybluemix.net)
- Official Tutorials (https://github.com/Trusted-AI/AIX360/tree/master/examples)
- Video Tutorial (https://www.youtube.com/watch?v=TGPHPCg_zKA)

What Do You Think?

Join Our Telegram Group. Be part of an engaging online community. Join Here (https://t.me/joinchat/NJLxnhZB7GkX3CPvjs9QGQ).

AISHWARYA VERMA (HTTPS://ANALYTICSINDIAMAG.COM/AUTHOR/AISHWARYA-VERMAANALYTICSINDIAMAG-COM/)

in(https://www.linkedin.com/in/aishwarya-verma-a46a4b174/)

A data science enthusiast and a post-graduate in Big Data Analytics. Creative and organized with an analytical bent of mind.

OUR UPCOMING EVENTS

SKILLUP 2021 | Data Science Education Fair | 22-23rd April | Register here>> (https://skillup.analyticsindiasummit.com/)

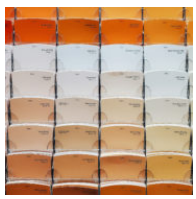
Rising 2021 (https://rising.analyticsindiasummit.com/) | Women in AI Conference | May 21 & 22 | Virtual

RELATED POSTS

DEVELOPERS CORNER (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/DEVELOPERS_CORNER/)

Hands-on Guide to Interpret Machine Learning with SHAP (https://analyticsindiamag.com/hands-on-guide-to-interpret-machine-learning-with-shap/)

06/03/2021 · 6 MINS READ



(https://analytics on-guide-to-interpret-machine-learning-with-shap/)

DEVELOPERS CORNER (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/DEVELOPERS_CORNER/)

Top 8 Indian Open-Source Projects Of 2020 (https://analyticsindiamag.com/top-8-indian-open-source-projects-of-2020/)

31/12/2020 · 4 MINS READ

(https://analytics 8-indian-open-source-projects-of-2020/)

OPINIONS (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/ARTICLES/)

A Greater Foundation For The Triumph Of Deep Learning With XAI (https://analyticsindiamag.com/a-greater-foundation-for-the-triumph-of-deep-learning-

(https://analytics greater-foundation-for-the-triumph-of-deep-learning-with-xai/)

Scroll To Top

21/12/2020 · 5 MINS READ

DEVELOPERS CORNER (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/DEVELOPERS_CORNER/)

Hands-On Guide To Natural language Processing Using Spacy (https://analyticsindiamag.com/nlp-deep-learning-nlp-framework-nlp-model/)

(https://analytics
deep-learning-
nlp-framework-
nlp-model/)

16/12/2020 · 6 MINS READ

OPINIONS (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/ARTICLES/)

Does India Have The Infrastructure To Implement Policy Recommendations To Avoid Algorithmic Bias (https://analyticsindiamag.com/does-india-have-the-infrastructure-to-implement-policy-recommendations-to-avoid-algorithmic-bias/)

(https://analytics
india-have-the-
infrastructure-
to-implement-
policy-
recommendations
to-avoid-
algorithmic-
bias/)

08/12/2020 · 4 MINS READ

DEVELOPERS CORNER (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/DEVELOPERS_CORNER/)

Top Milestones On Explainable AI In 2020 (https://analyticsindiamag.com/top-milestones-on-explainable-ai-in-2020/)

(https://analytics
milestones-on-
explainable-ai-
in-2020/)

25/11/2020 · 3 MINS READ

Guide To (6D Object Pose Estimation Using PoseCNN (https://analyticsindiamag.com/guide-to-6d-object-pose-estimation-using-posecnn/)

BY MOHIT MAITHANI (HTTPS://ANALYTICSINDIAMAG.COM/AUTHOR/MOHIT-MAITHANIANALYTICSINDIAMAG-COM/)
28/01/2021

PoseCNN(Convolutional Neural Network) is an end to end framework for 6D object pose estimation, It calculates the 3D translation (https://analyticsindiamag.com/guide-to-nvidia-imaginaire-gan-library-in-python/) of the object by localizing the mid of the image and predicting its distance from the camera, and the rotation is calculated by relapsing to a quaternion representation. PoseCNN is papered (https://arxiv.org/pdf/1711.00199.pdf)by Yu Xiang (https://yuxng.github.io/), Tanner Schmidt (https://homes.cs.washington.edu/~tws10/), Venkatraman Narayanan (https://www.cs.cmu.edu/~venkatrn/), and Dieter Fox (https://homes.cs.washington.edu/~fox/) in collaboration with Nvidia research. They also discussed a novel loss function that can help PoseCNN to handle symmetrical objects from images. They created a custom dataset YCB video dataset (https://paperswithcode.com/sota/6d-pose-estimation-on-ycb-video), which gives 6D poses of 21 objects in 92 videos with almost 133k frames for producing their results. PoseCNN is able to handle symmetrical objects pretty well and can do certain pose estimation using only a single image as an input.

-02:18

Network Architecture

The PoseCNN network contains two stages; the first stage is 13 CNN layers and four max-pooling layers, which helps extract feature maps with different input image resolution. The first stage is the primary backbone of the network.

The second stage is all about the embedding step that uses high feature maps generated by the first stage into low-dimensional features. After that network performs three different tasks and is trained to do specifically three tasks:

1. Semantic labeling.
2. 3D translation estimation.
3. 3D rotation regression.

1. Semantic labeling

Semantic labeling detects objects in images, where on the other hand network classifies each image input pixels into an object class. In comparison with the 6D pose estimation technique that leverages object detection using a bounding box, semantic labeling gives more information about the objects in the image and can handle occlusions better.

It takes two feature maps with dimensions 512 as inputs to the network, as shown in the above figure. The resolution is ⅛ and 1/16 of the original input image. It first reduces dimensions of the two features to 64 using the CNN layer. Then it doubles the resolution of that 1/16 feature map by using another deconvolutional layer. After that, another two feature map and deconvolution layer is used to increase the resolution of input by 8x. Finally, the convolutional layer produces a semantic labeling score for image pixels.

Remember, in training, a softmax cross-entropy is used, and in testing, the softmax function is used to predict image pixels class.

2. 3D translation estimation

3D translation localize the 2D object center in the image to estimate the object distance from the camera

3. 3D rotation regression

The lower part of the above architecture diagram shows the 3D rotation regression method. In this researchers tried to use the Hough voting layer object detection bounding box to predict two RoI pooling layers to crop and pool the feature of the image by generating the first stage of the network for 3D regression.

About layers, 3D rotation regression uses pooled feature map by integrating into three fully connected layers. The first two FC layers have dimensions 4096, and the last FC layer have 4 x n (n=number of object classes)

Dataset

The dataset used for this approach is the YCB dataset, it consists of 80 videos for train, and 2949 key features are extracted from the 12 test videos.

Implementation

It is trained and tested on Ubuntu 16.04 with PyTorch 0.41+ and CUDA 9.1

1. Install PyTorch (https://pytorch.org/)
2. Install Eigen from Github here (https://github.com/eigenteam/eigen-git-mirror)
3. Install Sophus from Github here (https://github.com/yuxng/Sophus)

```
git clone https://github.com/NVlabs/PoseCNN-PyTorch.git
(https://github.com/NVlabs/PoseCNN-PyTorch.git)
pip install -r requirement.txt
git submodule update --init --recursive
##Compile the new layers under $ROOT/lib/layers
cd $ROOT/lib/layers
sudo python setup.py install
##Compile cython
cd ..
cd $ROOT/lib/utils
python setup.py build_ext --inplace
##compile the ycb_render in $ROOT/ycb_render
cd ..
cd $ROOT/ycb_render
sudo python setup.py develop
```

Download

- Download 3D models of YCB Objects from here (https://drive.google.com/file/d/1PTNmhd-eSq0fwSPv0nvQN8h_scR1v-UJ/view?usp=sharing). And Save it under $ROOT/data.
- Download pre-trained checkpoints from here (https://drive.google.com/file/d/1-ECAkkTRfa1jJ9YBTzf04wxCGw6-m5d4/view?usp=sharing) and similarly save it under $ROOT/data.
- Real-world images with pose annotations for 20 YCB objects can be downloaded from here (https://drive.google.com/file/d/1cQH_dnDzyrI0MWNx8st4lht_q0F6cUrE/view?usp=sharing) (53Gb).

Running the demo

1. Download 3D models and our pre-trained checkpoints and setup environment.
2. run the following command

```
./experiments/scripts/demo.sh
```

Train and Test on YCB- dataset

First, download the YCB-Video dataset from here (https://rse-lab.cs.washington.edu/projects/posecnn/) and then create a symlink for the YCB-Video dataset using below command:

```
cd $ROOT/data/YCB_Video
 ln -s $ycb_data data
```

```
Let's Train and test on the YCB-Video dataset
cd $ROOT
 # multi-gpu training, use 1 GPU or 2 GPUs ./experiments/scripts/ycb_video_train.sh
 # testing, $GPU_ID can be 0, 1, etc.
 ./experiments/scripts/ycb_video_test.sh $GPU_ID
```

Conclusion

We learned the new method for object pose estimation, PoseCNN decouples the estimation of 3D rotation and translation. It localizes the object center and predicts the center distance of the image. To learn more you can follow given below resources:

Scroll To Top

- PoseCNN (GitHub) (https://github.com/yuxng/PoseCNN)
- Research paper (https://arxiv.org/pdf/1711.00199[illegible])
- The YCB-Video Dataset ~ 265G (https://drive.google.com/file/d/1if4VoEXNx9W3XCn0Y7Fp15B4GpcYbyYi/view?usp=sharing)
- The YCB-Video 3D Models ~ 367M (https://drive.google.com/file/d/1gmcDD-5bkJfcMKLZb3zGgH_HUFbulQWu/view?usp=sharing)
- The YCB-Video Dataset Toolbox (GitHub) (https://github.com/yuxng/YCB_Video_toolbox)

MOHIT MAITHANI (HTTPS://ANALYTICSINDIAMAG.COM/AUTHOR/MOHIT-MAITHANIANALYTICSINDIAMAG-COM/)

(https://twitter.com/xaret_) (https://www.linkedin.com/in/mohitmaithani/)

Mohit is a Data & Technology Enthusiast with good exposure to solving real-world problems in various avenues of IT and Deep learning domain. He believes in solving human's daily problems with the help of technology.

RELATED POSTS

DEVELOPERS CORNER (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/DEVELOPERS_CORNER/)

Guide To Kornia: An OpenCV-inspired PyTorch Framework (https://analyticsindiamag.com/guide-to-kornia-an-opencv-inspired-pytorch-framework/)

(https://analytics
to-kornia-an-
opencv-
inspired-
pytorch-
framework/)

20/03/2021 · 6 MINS READ

DEVELOPERS CORNER (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/DEVELOPERS_CORNER/)

What Is Transformer-iN-Transformer? (https://analyticsindiamag.com/what-

(https://analytics
is-transformer-

Scroll To Top

is-transformer-in-transformer/) (https://analyticsindiamag.com/)

14/03/2021 · 3 MINS READ

in-transformer/)

NEWS (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/NEWS/)

DailyHunt Parent Acquires Cognirel Technologies To Develop Computer Vision Models (https://analyticsindiamag.com/dailyhunt-parent-acquires-cognirel-technologies-to-develop-computer-vision-models/)

(https://analytics
parent-acquires-
cognirel-
technologies-to-
develop-
computer-
vision-models/)

23/02/2021 · 2 MINS READ

DEVELOPERS CORNER (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/DEVELOPERS_CORNER/)

Guide To Real-time Object Detection Model Deployment Using Streamlit (https://analyticsindiamag.com/guide-to-real-time-object-detection-model-deployment-using-streamlit/)

(https://analytics
to-real-time-
object-
detection-
model-
deployment-
using-
streamlit/)

22/02/2021 · 7 MINS READ

DEVELOPERS CORNER (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/DEVELOPERS_CORNER/)

Guide to VISSL: Vision Library for Self-Supervised Learning (https://analyticsindiamag.com/guide-to-vissl-vision-library-for-self-supervised-learning/)

(https://analytics
to-vissl-vision-
library-for-self-
supervised-
learning/)

16/02/2021 · 4 MINS READ

DEVELOPERS CORNER (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/DEVELOPERS_CORNER/)

Guide to Panoptic Segmentation – A Semantic + Instance Segmentation Approach

(https://analytics
to-panoptic-

Scroll To Top

(https://analyticsindiamag.com/guide-to-panoptic-segmentation-a-semantic-instance-segmentation-approach/)

05/02/2021 · 6 MINS READ

DEVELOPERS CORNER (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/DEVELOPERS_CORNER/)

Step-by-step Guide For Image Classification Using ML.NET (https://analyticsindiamag.com/step-by-step-guide-for-image-classification-using-ml-net/)

BY NIKITA SHILEDARBAXI (HTTPS://ANALYTICSINDIAMAG.COM/AUTHOR/NIKITAANALYTICSINDIAMAG-COM/)
28/01/2021

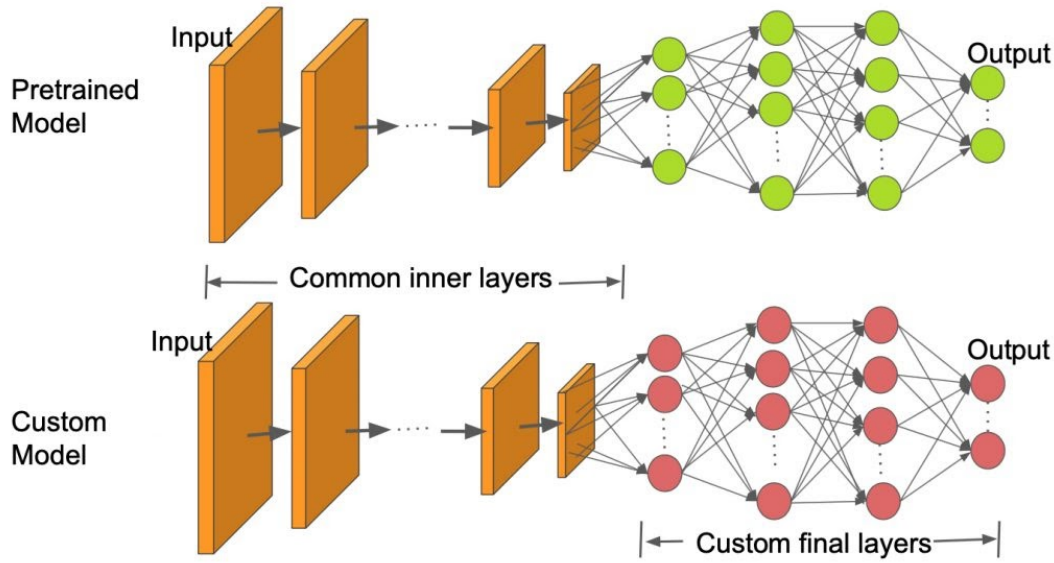
Table of contents

Scroll To Top

Image classification is a Computer Vision (https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e) task which falls into the category of Supervised Learning (https://towardsdatascience.com/a-brief-introduction-to-supervised-learning-54a3e3932590). We train a model to label an input image with one of the prescribed target classes based on the already labelled images of the training set. Here, we have a dataset having images of concrete surfaces. The task is to create a C# .NET Core console application which applies transfer learning, a pretrained TensorFlow model and ML.NET's Image classification API to identify the structures from the deck as cracked or uncracked.

In our previous article (https://analyticsindiamag.com/introduction-to-ml-net-a-machine-learning-framework-for-dotnet-developers/), we introduced ML.NET – a Microsoft Corporation (https://www.microsoft.com/en-in)'s project for .NET developers to accomplish Machine Learning tasks. Let us cover an important Deep Learning use case of ML.NET viz. image classification using the TensorFlow (https://www.tensorflow.org/) library and the concept of transfer learning.

Transfer Learning



Not aware of the concept of transfer learning? Refer to this (https://en.wikipedia.org/wiki/Transfer_learning#:~:text=Transfer%20learning%20(TL)%20is%20a,when%20trying%20to%20recognize%20trucks.) page before proceeding!

Image Classification API of ML.NET

The Image Classification API uses a low-level library called TensorFlow.NET (TF.NET). It binds .NET Standard framework (https://docs.microsoft.com/en-us/dotnet/standard/net-standard) with TensorFlow API (https://www.tensorflow.org/api_docs) in C#. It comes with a built-in high-level interface called TensorFlow.Keras (https://www.nuget.org/packages/TensorFlow.Keras/) .

Visit this (https://github.com/SciSharp/TensorFlow.NET) GitHub repository for detailed information on TF.NET.

Model training using transfer learning and the Image Classification API is a dual-phase process. The two phases included are as follows:

1. **Bottleneck phase**

The training set is loaded and the pixel values of those images are used as input for the frozen layers of the pre-trained model. The frozen layers consist of all the layers in the architecture up to the penultimate layer (also called the bottleneck layer). As no training actually occurs through these layers, they are referred to as 'frozen'. From these layers only, the pre-trained model learns to distinguish between the predefined target categories. The bottleneck phase occurs only once and its results can be cached for later usage.

2. **Training phase**

The output of the first phase is fed as an input to the ultimate layer of the model for its retraining. The number of times this happens is specified by the model parameters. Each iteration computes the model's accuracy and loss values depending on which further model optimizations can be carried out. At the end of the training phase, we get .zip and .pb formats of the model. It is preferable to use a .zip version in ML.NET-supported environments.

Dataset used

The SDNET2018 dataset used here is an annotated dataset comprising more than 56,000 images of cracked and non-cracked concrete walls, bridge decks and pavements.

Source (https://digitalcommons.usu.edu/all_datasets/48/): Maguire, Marc; Dorafshan, Sattar; and Thomas, Robert J., "SDNET2018: A concrete crack image dataset for machine learning applications" (2018)

Click here to download the .zip file (https://digitalcommons.usu.edu/cgi/viewcontent.cgi?filename=2&article=1047&context=all_datasets&type=additional) of the data.

The dataset has three subdirectories each containing images for one of the three types of structures:

D – for bridge decks

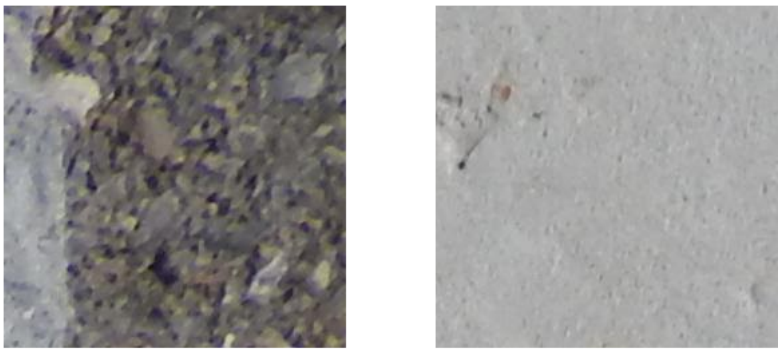
W – for walls

P – for pavements

For each of the above subdirectories, there is the further splitting of cracked and uncracked surfaces' images into two subdirectories with prefix 'C' and 'U' respectively.

Here, we are using only the 'D' subdirectory i.e. bridge deck images.

Sample cracked images



Sample uncracked images

Model architecture used

We have used a part of the 101-layer variant of the Residual Network (ResNet) v2 model whose original version takes 224*224 dimensional images and classifies them into appropriate categories.

Visit this (https://neurohive.io/en/popular-networks/resnet/) page to understand the model in detail.

Prerequisites for the implementation

- Use Visual Studio (https://visualstudio.microsoft.com/downloads/) 2019 or higher version
- Or use Visual Studio 2017 version 15.6 or higher with the NET Core cross-platform development workload installed

Create your C# .NET Core console application and then install the Microsoft.ML NuGet Package. Click here (https://www.nuget.org/packages/Microsoft.ML/) for its installation.

Open the Program.cs file and replace the 'using' statements with the following ones:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.IO;
using Microsoft.ML;
using static Microsoft.ML.DataOperationsCatalog;
using Microsoft.ML.Vision;
```

Data Preparation

Unzip the 'D' subdirectory and copy it into your project directory.

Define the image data schema below the 'Program' class by creating a class, say 'ImgData' as follows:

```
class ImgData
{
    //path of the image file
    public string ImgPath { get; set; }
   //category to which the image in ImgPath belongs to
    public string Label { get; set; }
}
```

Define the input data schema by creating a class say InputData as follows:

```
class InputData
{
   public byte[] Img { get; set; } //byte representation of image
   public UInt32 LabelKey { get; set; } //numerical representation of label
   public string ImgPath { get; set; } //path of the image
    public string Label { get; set; }
}
```

From the InputData class, 'Img' and 'LabelKey' properties will be used training and prediction purposes. 'ImgPath' and 'Label' columns have been included just to access the original file name and text representation of labels.

Define the output schema by creating a class, for example, Output as follows:

```
class Output
{
    public string ImgPath { get; set; } //path of the image
    public string Label { get; set; } //target category
    public string Pred { get; set; } //predicted label
}
```

'ImgPath' and 'Label' properties here play the same roles as in InputData class. Only 'Pred' property is used for prediction.

Creating workspace directory

If training and validation data do not not change frequently, cache the bottleneck values to be used for further runs. To store those values and .pb version of the model, create a directory say 'workspace' in your project.

Note: .pb stands for protobuf. In TensorFlow, .pb file is required to run a trained model. It consists of graph definition and weights of the model.

Path definitions and variable initialization

Inside the Main method, define the path location of your assets, computed bottleneck values and .pb version of the model.

```
var projectDir = Path.GetFullPath(Path.Combine(AppContext.BaseDirectory,
"../../../"));
var workspace = Path.Combine(projectDir, "workspace");
var assets = Path.Combine(projectDir, "assets");
```

Instantiate MLContext (https://docs.microsoft.com/en-us/dotnet/api/microsoft.ml.mlcontext?view=ml-dotnet) class.

```
MLContext myContext = new MLContext();
```

Data Loading

Create LoadImagesFromDirectory utility method below the Main method to format the data into a list of 'ImgData' class' objects since we have data distributed in two subdirectories (C-prefixed and U-prefixed).

```
public static IEnumerable<ImgData> LoadImagesFromDirectory(string folder, bool
useFolderNameAsLabel = true)
 {
     //get all file paths from the subdirectories
     var files = Directory.GetFiles(folder, "*",searchOption:
     SearchOption.AllDirectories);
     //iterate through each file
     foreach (var file in files)
     {
//Image Classification API supports .jpg and .png formats; check img formats
         if ((Path.GetExtension(file)!=".jpg") &&
          (Path.GetExtension(file)!=  ".png"))
             continue;
         //store filename in a variable, say 'label'
         var label = Path.GetFileName(file);
         /* If the useFolderNameAsLabel parameter is set to true, then name
            of parent directory of the image file is used as the label. Else label
is expected to be the file name or a a prefix of the file name. */
         if (useFolderNameAsLabel)
             label = Directory.GetParent(file).Name;
         else
         {
             for (int index = 0; index < label.Length; index++)
             {
                 if (!char.IsLetter(label[index]))
                 {
                     label = label.Substring(0, index);
                     break;
                 }
             }
         }
         //create a new instance of ImgData()
         yield return new ImageData()
         {
             ImagePath = file,
             Label = label
         };
     }
 }
```

Note: When 'yield return' statement is reached in a iterator methor in C# code, expression following it is returned and current code location is retained. The next time you call that function, execution restarts from that location only.

Get the list of images used for training using LoadImagesFromDirectory method.

```
IEnumerable<ImgData> imgs = LoadImagesFromDirectory(folder: assetsRelativePath,
useFolderNameAsLabel: true);
```

Scroll To Top

Load those images into an IDataView (https://docs.microsoft.com/en-us/dotnet/api/microsoft.ml.idataview) using LoadFromEnumerable (https://docs.microsoft.com/en-us/dotnet/api/microsoft.ml.dataoperationscatalog.loadfromenumerable)() method.

```
IDataView imgData = mlContext.Data.LoadFromEnumerable(imgs);
```

Data Preprocessing

Data gets loaded in the same order as it is read from the data subdirectories. Shuffle the data to add variance.

```
IDataView shuffle = mlContext.Data.ShuffleRows(imgData);
```

ML models expects numerical format of data. Preprocess the data by creating an EstimatorChain (https://docs.microsoft.com/en-us/dotnet/api/microsoft.ml.data.estimatorchain-1) having the MapValueToKey (https://docs.microsoft.com/en-us/dotnet/api/microsoft.ml.conversionsextensionscatalog.mapvaluetokey) and LoadRawImageBytes transforms.

```
 var preprocessingPipeline = my_Context.Transforms.Conversion.MapValueToKey
 /*takes the categorical value in the Label column, convert it to a numerical
KeyType value and store it in a new column called LabelKey*/
 (inputColumnName: "Label",
  outputColumnName: "LabelKey")
  /*take the values from the ImgPath column along with the imageFolder
    parameter to load images for training the model*/
  .Append(myContext.Transforms.LoadRawImageBytes(
   outputColumnName: "Img",
   imageFolder: assets,
   inputColumnName: "ImgPath"));
```

Use Fit (https://docs.microsoft.com/en-us/dotnet/api/microsoft.ml.data.estimatorchain-1.fit)() method to apply the shuffled data to the preprocessingPipeline EstimatorChain (https://docs.microsoft.com/en-us/dotnet/api/microsoft.ml.data.estimatorchain-1). Transform (https://docs.microsoft.com/en-us/dotnet/api/microsoft.ml.data.transformerchain-1.transform)() method is then applied to get an IDataView (https://docs.microsoft.com/en-us/dotnet/api/microsoft.ml.idataview) containing the pre-processed data.

```
IDataView preProcData = preprocessingPipeline.Fit(shuffle).Transform(shuffle);
```

Create train/validation/test datasets splits

```
 TrainTestData trainSplit = myContext.Data.TrainTestSplit(data: preProcData,
testFraction: 0.3);
 TrainTestData validationTestSplit =
myContext.Data.TrainTestSplit(trainSplit.TestSet);
```

testFraction: 0.3 means that 30% of the whole data is used as validation set while rest of the 70% as train set. From the validation set, 90% data is used for validation and remaining 10% for testing.

Create IDataView of each of the splits.

```
 IDataView trainSet = trainSplit.TrainSet;
 IDataView validationSet = validationTestSplit.TrainSet;
 IDataView testSet = validationTestSplit.TestSet;
```

Define model training pipeline

Store required and optional parameters of ImageClassificationTrainer (https://docs.microsoft.com/en-us/dotnet/api/microsoft.ml.vision.imageclassificationtrainer)

```
var classifierOptions = new ImageClassificationTrainer.Options()
{
    //input column for the model
    FeatureColumnName = "Image",
    //target variable column
    LabelColumnName = "LabelAsKey",
    //IDataView containing validation set
    ValidationSet = validationSet,
    //define pretrained model to be used
    Arch = ImageClassificationTrainer.Architecture.ResnetV2101,
    //track progress during model training
    MetricsCallback = (metrics) => Console.WriteLine(metrics),
    /*if TestOnTrainSet is set to true, model is evaluated against
      Training set if validation set is not there*/
    TestOnTrainSet = false,
   //whether to use cached bottleneck values in further runs
    ReuseTrainSetBottleneckCachedValues = true,
    /*similar to ReuseTrainSetBottleneckCachedValues but for validation
      set instead of train set*/
    ReuseValidationSetBottleneckCachedValues = true
};
```

Define the EstimatorChain (https://docs.microsoft.com/en-us/dotnet/api/microsoft.ml.data.estimatorchain-1) training pipeline

```
var trainingPipeline =
mlContext.MulticlassClassification.Trainers.ImageClassification(classifierOptions).
Append(mlContext.Transforms.Conversion.MapKeyToValue("PredictedLabel"));
```

Fit the training data to the pipeline

```
ITransformer trainedModel = trainingPipeline.Fit(trainSet);
```

Create utility method to display predictions made by the model

```
private static void OutputPred(Output pred)
{
    string imgName = Path.GetFileName(pred.ImgPath);
    Console.WriteLine($"Image: {imgName} | Actual Label: {pred.Label} |
    Predicted Label: {pred.PredictedLabel}");
}
```

Make prediction for a single image

Scroll To Top

```
 public static void ClassifyOneImg(MLContext myContext, IDataView data,
ITransformer trainedModel)
 {
     PredictionEngine<InputData, Output> predEngine =
myContext.Model.CreatePredictionEngine<InputData, Output>(trainedModel);
     InputData image = myContext.Data.CreateEnumerable<InputData>
(data,reuseRowObject:true).First();
     Output prediction = predEngine.Predict(image);
     //print predicted value
     Console.WriteLine("Prediction for single image");
     OutputPred(prediction);
 }
```

Run ClassifyOneImg() in your application

```
ClassifyOneImg(myContext, testSet, trainedModel);
```

Make predictions for multiple images

```
 public static void ClassifyMultiple(MLContext myContext, IDataView data,
ITransformer trainedModel)
 {
     IDataView predictionData = trainedModel.Transform(data);
     IEnumerable<Output> predictions =
     myContext.Data.CreateEnumerable<Output>(predictionData, reuseRowObject:
      true).Take(20); //20 images
     Console.WriteLine("Prediction for multiple images");
     foreach (var p in predictions)
     {
         OutputPred(p); //print predicted value of each image
     }
 }
```

Run ClassifyMultiple() in your application

```
ClassifyMultiple(myContext, testSet, trainedModel);
```

Run your console application

Output

The output will look something like this:

Bottleneck phase:

```
Phase: Bottleneck Computation, Dataset used:      Train, Image Index: 279
Phase: Bottleneck Computation, Dataset used:      Train, Image Index: 280
Phase: Bottleneck Computation, Dataset used: Validation, Image Index:   1
Phase: Bottleneck Computation, Dataset used: Validation, Image Index:   2
```

Training phase:

```
Phase: Training, Dataset used: Validation, Batch Processed Count:   6,
Epoch:  21, Accuracy:  0.6757613
Phase: Training, Dataset used: Validation, Batch Processed Count:   6,
Epoch:  22, Accuracy:  0.7446856
Phase: Training, Dataset used: Validation, Batch Processed Count:   6,
Epoch:  23, Accuracy:  0.7716660
```

Output of classification:

Scroll To Top

```
Prediction for single image
Image: 7001-220.jpg | Actual Value: UD | Predicted Value: UD
Prediction for multiple images
Image: 7001-220.jpg | Actual Value: UD | Predicted Value: UD
Image: 7001-163.jpg | Actual Value: UD | Predicted Value: UD
Image: 7001-210.jpg | Actual Value: UD | Predicted Value: UD
Image: 7004-125.jpg | Actual Value: CD | Predicted Value: UD
```


Ways to improve model's performance

- Use more data from the dataset instead of just sticking to bridge deck images
- Try using some other model architecture
- Try varying the values of some hyperparameters
- Perform data augmentation
- Train for more time by incrementing the number of epochs

References

Following are the sources used for the above-explained code and its implementation procedure:

- GitHub (https://github.com/dotnet/machinelearning-samples/tree/master/samples/csharp/getting-started/DeepLearning_ImageClassification_Binary)
- Microsoft Tutorial (https://docs.microsoft.com/en-us/dotnet/machine-learning/tutorials/image-classification-api-transfer-learning)

NIKITA SHILEDARBAXI (HTTPS://ANALYTICSINDIAMAG.COM/AUTHOR/NIKITAANALYTICSINDIAMAG-COM/)

A zealous learner aspiring to advance in the domain of AI/ML. Eager to grasp emerging techniques to get insights from data and hence explore realistic Data Science applications as well.

RELATED POSTS

DEVELOPERS CORNER (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/DEVELOPERS_CORNER/)

Comprehensive Guide To Demand Forecasting Using ML.NET (https://analyticsindiamag.com/comprehensive-guide-to-demand-forecasting-using-ml-net/)

(https://analytics guide-to- demand- forecasting- using-ml-net/)

01/02/2021 · 7 MINS READ

DEVELOPERS CORNER (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/DEVELOPERS_CORNER/)

Guide to Product

Scroll To Top

Recommendation Using ML.NET (https://analyticsindiamag.com/guide-to-product-recommendation-using-ml-net/)

30/01/2021 · 4 MINS READ

(https://analytics
to-product-
recommendation-
using-ml-net/)

DEVELOPERS CORNER (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/DEVELOPERS_CORNER/)

The Evolution of ImageNet for Deep Learning in Computer Vision (https://analyticsindiamag.com/imagenet-and-variants/)

13/11/2020 · 6 MINS READ

(https://analytics
and-variants/)

DEVELOPERS CORNER (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/DEVELOPERS_CORNER/)

A Tutorial On Google Teachable Machine For Object Classification Without Coding (https://analyticsindiamag.com/a-tutorial-on-google-teachable-machine-for-object-classification-without-coding/)

22/10/2020 · 4 MINS READ

(https://analytics
tutorial-on-
google-
teachable-
machine-for-
object-
classification-
without-coding/)

DEVELOPERS CORNER (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/DEVELOPERS_CORNER/)

Complete Guide To ShuffleNet V1 With Implementation In Multiclass Image Classification (https://analyticsindiamag.com/complete-guide-to-shufflenet-v1-with-implementation-in-multiclass-image-classification/)

13/10/2020 · 9 MINS READ

(https://analytics
guide-to-
shufflenet-v1-
with-
implementation-
in-multiclass-
image-
classification/)

DEVELOPERS CORNER (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/DEVELOPERS_CORNER/)

Scroll To Top

8 Important Hacks for Image Classification Models One Must Know (https://analyticsindiamag.com/8-important-hacks-for-image-classification-models-one-must-know/)

(https://analytics
important-
hacks-for-
image-
classification-
models-one-
must-know/)

07/10/2020 · 3 MINS READ

DEVELOPERS CORNER (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/DEVELOPERS_CORNER/)

Porcupine: A Compiler That Translates Plain Text To Encrypted Code On The Go (https://analyticsindiamag.com/porcupine-a-compiler-that-translates-plain-text-to-encrypted-code-on-the-go/)

BY SHRADDHA GOLED (HTTPS://ANALYTICSINDIAMAG.COM/AUTHOR/SHRADDHA-GOLEDANALYTICSINDIAMAG-COM/)
28/01/2021

→ The team claimed Porcupine had increased the processing speed by 51 percent compared to hand-optimised and heuristic-driven code.

Scroll To Top

Homomorphic encryption, a 'privacy-preserving technique (https://analyticsindiamag.com/) that performs computation directly on the encrypted data. However, this encryption technique faces significant performance and programmability challenges.

Researchers from Facebook, New York University, and Stanford University have created a synthesising compiler caller Porcupine to unleash the true potential of homomorphic encryption technique. This compiler can translate plain text to encrypted code on the go.

The team claimed the compiler had increased the processing speed by 51 percent compared to hand-optimised and heuristic-driven code.

What Is Homomorphic Encryption

The first step while working with encrypted data is decryption. However, it leaves the data vulnerable to attack, defeating the very purpose of encryption. In such a case, an ideal solution would be to have a system that could process information without compromising privacy and security.

Dr Craig Gentry, the creator of the first Fully Homomorphic Encryption scheme, compares (https://www.forbes.com/sites/bernardmarr/2019/11/15/what-is-homomorphic-encryption-and-why-is-it-so-transformative/) the technique a glovebox, a sealed container where you can put your hands to work the material inside, but can't take anything out of it.

Homomorphic encryption (https://analyticsindiamag.com/the-politics-of-end-to-end-encryption/) is beneficial when data needs to be outsourced to cloud environments and other partners for research and analytics purposes. The encryption technique makes it possible to analyse data without jeopardising the privacy and have applications in financial services, healthcare, and information technology. Unlike other modern encryption models, the technique doesn't yield to the decryption bids of quantum (https://analyticsindiamag.com/indian-firms-need-quantum-secure-key-distribution-to-prevent-future-attacks-says-cto-of-qnu-labs/) computers.

That said, homomorphic encryption has its share of shortcomings. The technique, being a slowpoke, is not practical in many applications. Plus, several performance overheads and compilation challenges stand in the way of its widespread adoption.

How Does Porcupine Help?

Many scientists are working towards fixing the drawbacks of homomorphic encryption (https://analyticsindiamag.com/implementing-encryption-and-decryption-of-data-in-python/) technique. However, not much work has been done in the automatic compilation of homomorphic encryption kernels.

On a broader scale, homomorphic encryption has three major compilation challenges:

- It supports only a limited set of a single instruction, multiple data (SIMD)-type operators
- It uses long-vector operands

Scroll To Top

- If the ciphertext noise growth is not controlled, it may result in the failure of the decryption process.

In this encryption technique, a programmer is first required to break the input kernel into SIMD addition, multiplication, and rotation instructions. This makes kernel implementation complex. Currently, these kernels are handwritten by experts, making it difficult to scale up. As a result, there is a need for automated compiler support for privacy-preserving computation.

To overcome the challenges, the team has presented an optimising compiler Porcupine. Given a plaintext code, Porcupine automatically synthesises a homomorphic encryption code using a component called Quill. This component helps search for 'verifiably correct' kernels and minimises kernel's costs such as latency and noise accumulation. With Porcupine and Quill, we get a synthesis procedure that automates the mapping and scheduling process of plaintext kernels to the homomorphic encryption (https://analyticsindiamag.com/top-3-reasons-why-your-data-needs-encryption/) instructions.

The Porcupine compiler was tested using a range of image processing and linear algebra problems. This compiler was evaluated using nine kernels to demonstrate that it could successfully translate plain text specification to homomorphic encryption-equivalent implementations. It was found that, over the handwritten-baselines, Porcupine delivered 51 percent faster with a geometric mean of 11 percent across the test kernels.

For smaller programs, Porcupine was able to find the same optimised implementation as the handwritten ones. For larger and complex programs, Porcupine could discover application-specific optimisation involving separable filters.

Wrapping Up

By automating the tasks, Porcupine frees up application designers to focus on other priorities. The researchers believe the homomorphic encryption technique is a rapidly developing area with a lot of scope for improvements, including enhancing compilation infrastructures such as Porcupine.

Read the full paper here (https://arxiv.org/pdf/2101.07841.pdf).

SHRADDHA GOLED (HTTPS://ANALYTICSINDIAMAG.COM/AUTHOR/SHRADDHA-GOLEDANALYTICSINDIAMAG-COM/)

I am a journalist with a postgraduate degree in computer network engineering. When not reading or writing, one can find me doodling away to my heart's content.

(https://t.me/share/url?&text=&url=https://analyticsindiamag.com/porcupine-a-compiler-that-translates-plain-text-to-encrypted-code-on-the-go/)
(https://share.flipboard.com/bookmarklet/popout?v=2&title=&url=https://analyticsindiamag.com/porcupine-a-compiler-that-translates-plain-text-to-encrypted-code-on-the-go/)

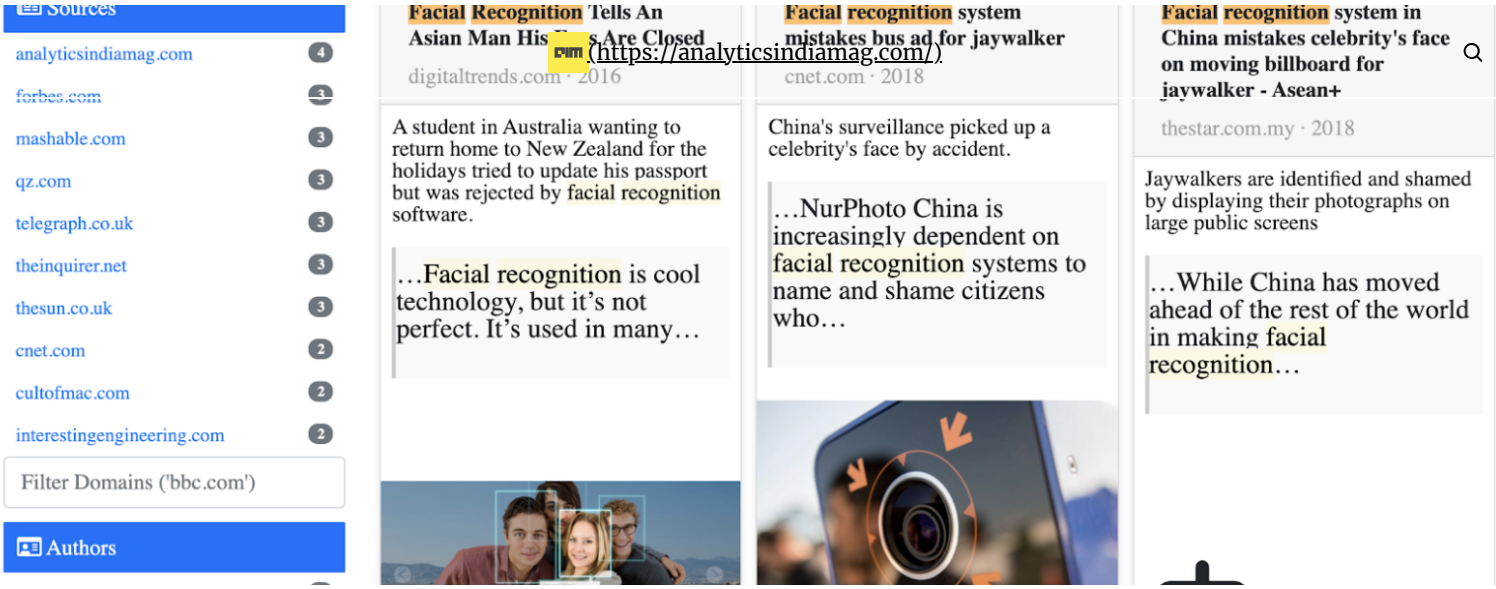
DEVELOPERS CORNER (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/DEVELOPERS_CORNER/)

What Is AI Incident Database? (https://analyticsindiamag.com/what-is-ai-incident-database/)

BY SEJUTI DAS (HTTPS://ANALYTICSINDIAMAG.COM/AUTHOR/SEJUTI-DASANALYTICSINDIAMAG-COM/)
27/01/2021

→ The PAI has introduced the AI Incident Database to help practitioners figure out things that can go wrong when AI systems are deployed.

Scroll To Top



Today, businesses and government organisations are increasingly deploying intelligent systems to safety-critical problem areas (https://analyticsindiamag.com/model-explainability-validates-ai-for-safety-critical-systems-says-prashant-rao-mathworks-india/) such as healthcare, credit scoring, law enforcement, aircraft control, and corporate recruitment. Failures of such systems pose severe risks to life and expose the limits of intelligent systems deployed in real-world situations. The wrongful arrest of Robert Williams (https://www.nytimes.com/2020/06/24/technology/facial-recognition-arrest.html) due to a flawed facial recognition system is a good case in point.

Experts believe AI practitioners should be aware of past failures of intelligent systems to avoid such fiascos. To that end, the Partnership on AI (PAI) (https://www.partnershiponai.org/), a nonprofit organisation established to outline best practices in AI technologies, has introduced the AI Incident Database (AIID) (https://incidentdatabase.ai/). A compelled repository of AI failures (https://analyticsindiamag.com/biggest-ai-goof-ups-that-made-headlines-in-2020/), AIID helps practitioners figure out what can go wrong when the system is deployed. Simply put, the database makes it easy for AI practitioners to learn from previous mistakes.

Led by Sean McGregor (https://www.linkedin.com/in/seanbmcgregor/), the technical lead of IBM Watson AI XPRIZE, AI Incident Database provides an infrastructure supporting AI best practices, a dataset of more than one thousand incidents, and an architecture for building research products.

Also Read: Are Blockchains More Secure Than Distributed Databases? (https://analyticsindiamag.com/)

Nuts & Bolts

Scroll To Top

The AI Incident Database (https://www.partnershiponai.org/aiincidentdatabase/) (AIID) catalogues more than 1,000 publicly available incident reports, (https://analyticsindiamag.com/) including documents and reports from the academic press. According to the paper released by McGregor, these AI failure reports serve a range of purposes, starting from providing multiple viewpoints on incidents, to the number of publications types that double as a proxy for interest in the incident. Additionally, sampling multiple reports per incident would provide more comprehensive coverage of the incident, which increases the practitioners' chances to discover relevant incidents.

Explaining the process, McGregor told media (https://bdtechtalks.com/2021/01/14/ai-incident-database/) that considering AI systems learn to operate from their training data, it can easily change its conduct based on such data. Thus, such AI-based safety-critical systems can map new possibilities for failure.

According to McGregor, most incidents submitted revolve around Ethical AI, (https://analyticsindiamag.com/eu-fundamental-rights-agency-issues-report-on-ai-ethical-considerations/) especially facial recognition systems, followed by failures in autonomous cars and trading algorithms that either cause substantial damage or put lives at risk.

AI practitioners can even search the database based on keywords, source, and authors involved, to get a 360-degree view. For instance — searching for 'facial recognition (https://analyticsindiamag.com/can-this-ai-filter-protect-human-identities-from-facial-recognition-system/)' will bring up 98 reports of AI incidents involving failures or problems related to automatic face recognition, biometric identification, identity verification etc. The search can be further refined based on the requirements.

The database's outline has been massively inspired by the 'Aviation Accident Reports (https://www.ntsb.gov/_layouts/ntsb.aviation/index.aspx)' — a shared database critically designed for managing flight safety by analysing the aircraft's past incidents. McGregor said, the AI Incident Database will help users manage the safety of the AI systems deployed in the real world. The AIID is a collection of web applications that interfaces with a MongoDB document database storing incident report text and metadata.

Also Read: Why Do Facial Recognition Systems Still Fail (https://analyticsindiamag.com/why-facial-recognition-systems-fail-stanford-report/)

Applications

Such pragmatic coverage of AI incidents can help AI practitioners discover and understand past experiences and create more possibilities in deploying AI systems in real-world applications. McGregor explained some of the critical areas, including deploying AI-powered recommendation systems or integrating ML systems to reduce financial and compliance risks (https://analyticsindiamag.com/how-effective-are-bug-bounty-programs-as-security-compliance-strategies/). Engineers can also use AIID to learn more about the environment their systems are deployed within, and researchers can understand the AI systems' safety and fairness.

Identifying AI failures by the PAI members started in 2018. However, nobody kept a record of it until now.

McGregor has even open-sourced the project on GitHub (https://github.com/PartnershipOnAI/aiid), where he has welcomed industry users to improve its capabilities and build taxonomies and data summaries in the AIID codebase. Making the database shareable will persuade technology companies to evaluate the bad outcomes before implementation. In due time, McGregor hopes the database will develop into community-owned infrastructure to help create beneficial intelligent systems for the greater common good.

Read the paper here (https://arxiv.org/pdf/2011.08512.pdf).

SEJUTI DAS (HTTPS://ANALYTICSINDIAMAG.COM/AUTHOR/SEJUTI-DASANALYTICSINDIAMAG-COM/)

Sejuti currently works as Senior Technology Journalist at Analytics India Magazine (AIM). Reach out at sejuti.das@analyticsindiamag.com

&url=https://analyticsindiamag.com/what-is-ai-incident-database/)

RELATED POSTS

PEOPLE (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/INTERVIEWS/)

Experiential Learning Is Not An Afterthought: Director, Computer Science & Engineering, BML Munjal University (https://analyticsindiamag.com/experiential-learning-is-not-an-afterthought-director-computer-science-engineering-bml-munjal-university/)

Scroll To Top

afterthought-director-computer-science-engineering-bml-munjal-university/) (https://analyticsindiamag.com/)

25/02/2021 · 7 MINS READ

DEVELOPERS CORNER (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/DEVELOPERS_CORNER/)

Top 8 Data Transformation Methods (https://analyticsindiamag.com/top-8-data-transformation-methods/)

(https://analytics 8-data-transformation-methods/)

22/01/2021 · 2 MINS READ

DEVELOPERS CORNER (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/DEVELOPERS_CORNER/)

Most Prominent Time Series Databases For Data Scientists (https://analyticsindiamag.com/most-prominent-time-series-databases-for-data-scientists/)

(https://analytics prominent-time-series-databases-for-data-scientists/)

20/01/2021 · 4 MINS READ

CAREERS (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/CAREERS/)

Top SQL Interview Questions For Data Scientists (https://analyticsindiamag.com/top-sql-interview-questions-for-data-scientists/)

(https://analytics sql-interview-questions-for-data-scientists/)

20/01/2021 · 4 MINS READ

STARTUPS (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/STARTUPS/)

Now, Companies Want To Go Digital As Early As Yesterday: Sudeep Srivastava, Appinventiv (https://analyticsindiamag.com/now-companies-want-to-go-digital-as-early-as-yesterday-sudeep-

(https://analytics companies-want-to-go-digital-as-early-as-yesterday-sudeep-

as early as yesterday Sudeep Srivastava (https://analyticsindiamag.com/as-early-as-yesterday-sudeep-srivastava-appinventiv/)

11/01/2021 · 4 MINS READ

CAREERS (HTTPS://ANALYTICSINDIAMAG.COM/CATEGORY/CAREERS/)

8 Latest Data Science Internship Openings One Must Apply Now (https://analyticsindiamag.com/8-latest-data-science-internship-openings-one-must-apply-now/)

08/01/2021 · 2 MINS READ

Scroll To Top