International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 03 | March - 2023 Impact Factor: 7.185 ISSN: 2582-3930

Machine Learning Framework to resolve Industrial Hassle

Mrs. Archana Kalia

VPM's Polytechnic ,Thane

Abstract: Common Manual Problem detected in any construction industry is "**Efficiently managing the labor force in different variety of sites".** Human forces have shown their managerial skills since ages. But then at times many skilled managers also fail in the process of management. But if the computer system is trained to do this particular job, at least 90% accuracy can be achieved and it can be further increased. This process of training can be carried away with the help of Machine learning implementation. This paper is trying to achieve the means of Machine learning implementation and using it to automate any machine which is commonly used in industries.

Key Words: Attributes, Preprocessing, Prediction, Valediction, Machine learning.

Introduction:

The idea of automation has been referred to any system that performs standardized, repetitive actions. Machine learning focuses on analyzing large data quantities, typically to make predictions. It is a later-stage development, where machines take in data on their own and then analyze it. It involves an entire category of technologies that provide activity or work without human involvement.

A simplified framework to machine learning includes the **five main areas of the machine learning process**:

1 - Data collection and preparation: It includes everything from choosing where to get the data, up to the point it is clean and ready for feature selection/engineering

2 - Feature selection and feature engineering: This includes all changes to the data from once it has been cleaned up to when it is ingested into the machine learning model

3 - Choosing the machine learning algorithm and training our first model: It involves in getting a "better than baseline" result upon which we can (hopefully) improve.

4 - Evaluating our model: This comprises the selection of the measure as well as the actual evaluation; seemingly a smaller step than others, but important to our end result

5 - Model tweaking, regularization, and hyper parameter tuning: This is where we iteratively go from a "good enough" model to our best effort.

6 - Prediction: To use the finalize model, to make prediction for the given problem statement.

Methodology

Step 1:

Deciding the attributes or characteristics of the given problem i.e automating a cement mixer.

Selecting attributes is similar to finding a needle in a haystack to solve our problem.

In this case, few of the given generalized features as our attributes are to be considered.

I. Age, Skill and experience of laborers
II. Absentee time, including late start and early quits
III. Non-working holidays
IV. Job complexity.
V. Job accessibility.
VI. Labor availability.
VII. Equipment utilization.
VIII. Local climate

While considering the above attributes, the problem statement can be broken down into several sub problems such as

i) Prediction of skilled person suitable for a particular job
ii) Prediction of job complexity with respect to availability of good quality equipment.
iii) Prediction of labor availability with respect to local climate.

Anyways we are discussing the major problem statement with respect to the above given features.

Step 2:

Collections of Real time data:

This has to be achieved in different construction companies where human beings are efficiently managing the workforce.

When building a data set, diversity of data has to be aimed. Usually companies are recommended to gather both internal and external data. The goal is to build a unique data set that will be hard for the competitors to copy. Machine learning applications require a large number of data points, but this doesn't mean the model has to consider a wide range of features. Meaningful data related to the project is required.

Now the data can be collected **based on questions** depending on the above attributes. For example, whether the particular person's age, skill and experience makes him eligible for the particular job scheduled in a particular site of construction. The answer can be 'YES' or 'No'.

Similarly other data to be checked which tends to answer to the questions of the above features.

Thus the data set is created with more than thousands of data within it.

Step 3:

Third step is to **Pre-process the existing data**.

What is Data pre-processing?

Data pre-processing is a process of cleaning the raw data i.e. the data is collected in the real world and is converted to a clean data set. In other words, whenever the data is gathered from different sources it is collected in a raw format and this data isn't feasible for the analysis.

Therefore, certain steps are executed to convert the data into a small clean data set, this part of the process is called as data pre-processing.

Requirements:

The data pre-processing is a process of cleaning the raw data into clean data, so that can be used to train the model. So, need for data pre-processing is to achieve good results from the applied model in machine learning and deep learning projects.

Most of the real-world data is messy, some of these types of data are:

1. **Missing data:** Missing data can be found when it is not continuously created or due to technical issues in the application.

2. **Noisy data:** This type of data is also called outliners, this can occur due to human errors (human manually gathering the data) or some technical problem of the device at the time of collection of data.

3. **Inconsistent data:** This type of data might be collected due to human errors (mistakes with the name or values) or duplication of data.

Three Types of Data

1. Numeric e.g. income, age
2. Categorical e.g. gender, nationality
3. Ordinal e.g. low/medium/high

How can data pre-processing be performed?

These are some of the basic pre - processing techniques that can be used to convert raw data.

1. **Conversion of data:** Machine Learning models can only handle numeric features, hence categorical and ordinal data must be somehow converted into numeric features.

2. **Ignoring the missing values:** Whenever missing data is encountered in the data set, the row or column of data can be removed depending on the need. This method is known to be efficient but it shouldn't be performed if there are a lot of missing values in the dataset.

3. **Filling the missing values:** Whenever missing data is encountered in the data set then the missing data is filled manually, most commonly the mean, median or highest frequency value is used.

Step 4:

Prediction of Algorithm-

This step can be done by categorizing the problem. This is a two-step process.

1. Categorize by input:

   - In case the data are labeled, it's a supervised learning problem.
   - In case of unlabeled data and a structure is required to be determined, it's an unsupervised learning problem.
   - If it is to optimize an objective function by interacting with an environment, it's a reinforcement learning problem.

2. Categorize by output.

- If the output of the model is a number, it's a regression problem.
- If the output of the model is a class, it's a classification problem.
- If the output of the model is a set of input groups, it's a clustering problem.

In the process of solving the given problem, it can be categorized as supervised learning since we have to detect the correct number of workforce that can be distributed to a particular site with its own features of work done based upon the other criteria.

Logistic Regression algorithm can be implemented for segmentation of labors and their management based on it.

It is advisable to **start with a very simplistic model** with minimal and most prominent set of features. This would help one to get started very quickly without spending time in exploring the correct and most appropriate features set. Many a times, lot of time is spent on identification of most appropriate features.

Plot learning curves can be implemented to measure how error (prediction vs. observed) varies with respect to some of the following:

- Adding more training examples. In simple words, collect more data sets.
- Adding more features
- Regularization parameters

Learning curves could very well help in examining the cases of high bias (under-fitting) or high variance (over-fitting).

Manual examination of the errors are to be completed that happened while testing the algorithm with cross-validation data set. This process would primarily help in identifying new features.

Step 5:

Split the data set into following three classes of data sets:

- Training data set
- Cross-validation data set
- Test data set

Training set: The training set is the material through which the computer learns how to process information. Machine learning uses algorithms to perform the training part. A set of data used for learning is to fit the parameters of the classifier.

Validation set: Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. A set of unseen data is used from the training data to tune the parameters of a classifier.

Test set: A set of unseen data used only to assess the performance of a fully-specified classifier.

Once the data is divided into the 3 given segments we can start the training process.

In a data set, a training set is implemented to build up a model, while a test (or validation) set is to validate the model built. Data points in the training set are excluded from the test (validation) set. Usually, a data set is divided into a training set, a validation set (some people use 'test set' instead) in each iteration, or divided into a training set, a validation set and a test set in each iteration.

Once the model is trained , the same trained model can be used to predict utilizing the testing data i.e. the unseen data. Once this is done , confusion matrix is developed further, this explains how well the model is trained.

A confusion matrix has 4 parameters, which are '**True positives**', **'True Negatives'**, **'False Positives'** and '**False Negative**'.

It is preferred that more the values in the True negatives and true positives, the more accurate model is developed. The size of the Confusion matrix completely depends upon the number of classes.

n=165	**Predicted: NO**	**Predicted: YES**
Actual: NO	50	10
Actual: YES	5	100

- **True positives:** These are cases in which we predicted TRUE and our predicted output is correct.
- **True negatives:** Predicted FALSE and predicted output is correct.
- **False positives:** Predicted TRUE, but the actual predicted output is FALSE.
- **False negatives:** Predicted FALSE, but the actual predicted output is TRUE.

✓ We can also find out the accuracy of the model using the confusion matrix.
✓ *Accuracy = (True Positives +True Negatives) / (Total number of classes)*
✓ i.e. for the above example: Accuracy = (100 + 50) / 165 = 0.9090 (90.9% accuracy)

Why do we use Resampling Methods?

The problem with applied machine learning is that unknown model is tried on. On a given predictive modeling problem, the ideal model is one that performs the best when making predictions on new data. When new data is not present, pretention with statistical tricks is carried over.

The train-test split and k-fold cross validation are called resampling methods. Resampling methods are statistical procedures for sampling a dataset and estimating an unknown quantity.

In the case of applied machine learning, industries are interested in estimating the skill of a machine learning procedure on unseen data. More specifically, the skill of the predictions made by a machine learning procedure.Once the estimated skill is achieved, the resampling method also gets over.

- If a train-test split is used, it means the split datasets and the trained model can be discarded
- If k-fold cross-validation is used, it means all of the trained models can be thrown away

They have served their purpose and are no longer needed. End users are ready to finalize the model.

Step 5:

Parameter Tuning

Once evaluation is done, it's possible that it is required to further improve the training in any way. This can be done by **tuning the parameters**. There were a few parameters that are implicitly assumed when training is achieved, and now is a good time to go back and test those assumptions and try other values.

Step 6:

Prediction:

Machine learning is using data to answer questions. So **Prediction**, or inference, is the step where we get to answer some questions. This is the point of all this work, where the value of machine learning is realized.

The finalize model is decided to make prediction for the given problem statement i.e the model will be able to judge how many skilled persons can be appointed to particular field of work in particular construction site.

This reduces the hectic job of a manager who tries his best to appoint the best of people in a site to get the utmost result and produces best result in terms of productivity.

Hence it is concluded that there are incredible opportunities for the construction companies to benefit from rapidly advancing automation, connectivity, and information technologies.

Implementation:

Cement mixer is one of the most critical machines in concrete technology. The automation of the mixer aims to create a cement mixer which helps to deliver concrete with desired features and quality without human intervention.

The properties and performance of mixer can be tailored to meet design requirements by varying the instructions set in the model.

Since the feature collected includes a small data set, Rule approach works best for precision.
Initially text analysis was done based on created feature set.

Given below is the rule based approach implemented in python:

```
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk import word_tokenize
import pandas as pd
import numpy as np

def preprocess(sentence):
   tokens = [word.lower() for word in
sentence.split()];
   #remove stopwords
   tokens = [i for i in tokens if i in
['in','off','down', 'on', 'up'] or not i in stoplist]
   #lemmatize verb words
   tokens = [wordnet_lemmatizer.lemmatize(i,
pos='v') for i in tokens]
   #lemmatize noun words
   tokens = [wordnet_lemmatizer.lemmatize(i,
pos='n') for i in tokens]
   new_tokens =[]

   #unigrams
   for token in tokens:
      new_tokens.append(token)

   #create bigrams
   for i in range(len(tokens)-1):
      token = '{}
{}'.format(tokens[i],tokens[i+1])
      new_tokens.append(token)

   return new_tokens

def predict_intent(tokens):
   for token in tokens:
      if token in ['start', 'kickstart', 'turn on',
'initiate', 'activate']:
         return 'start'
      if token in ['stop', 'shutdown', 'turn off',
'halt', 'end']:
         return 'stop'
      if token in ['set', 'initialize', 'fix']:
         return 'set'
      if token in ['increase', 'boost', 'step up']:
         return 'increase'
      if token in ['decrease', 'reduce', 'lower']:
         return 'decrease'
      if token in ['tilt', 'incline', 'lean']:
         return 'tilt'

#import stoplist that removes stop words
from a document

stoplist = stopwords.words('english')

#import wordnet lemmatizer
wordnet_lemmatizer = WordNetLemmatizer()

sentence = 'turn on the mixer'
tokens = preprocess(sentence)
print predict_intent(tokens)

sentence = 'start the mixer'
tokens = preprocess(sentence)
print predict_intent(tokens)

sentence = 'shutdown the mixer'
tokens = preprocess(sentence)
print predict_intent(tokens)

sentence = 'increase the speed of the mixer'
tokens = preprocess(sentence)
print predict_intent(tokens)
```

In the next step, speech to text recognition was done.
It is also provided with screen shot of the output done with commands.

```
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk import word_tokenize
import pandas as pd
import numpy as np
import speech_recognition as sr


def preprocess(sentence):
   tokens = [word.lower() for word in
sentence.split()];
   #remove stopwords
   tokens = [i for i in tokens if i in
['in','off','down', 'on', 'up'] or not i in stoplist]
   #lemmatize verb words
   tokens = [wordnet_lemmatizer.lemmatize(i,
pos='v') for i in tokens]
   #lemmatize noun words
   tokens = [wordnet_lemmatizer.lemmatize(i,
pos='n') for i in tokens]
   new_tokens =[]

   #unigrams
   for token in tokens:
      new_tokens.append(token)

   #create bigrams
   for i in range(len(tokens)-1):
      token = '{}
{}'.format(tokens[i],tokens[i+1])
      new_tokens.append(token)

   return new_tokens

def predict_intent(tokens):
   for token in tokens:
      if token in ['start', 'kickstart', 'turn on',
'initiate', 'activate']:
         return 'start'
      if token in ['stop', 'shutdown', 'turn off',
'halt', 'end']:
         return 'stop'
      if token in ['set', 'initialize', 'fix']:
         return 'set'
      if token in ['increase', 'boost', 'step up']:
         return 'increase'
      if token in ['decrease', 'reduce', 'lower']:
         return 'decrease'
      if token in ['tilt', 'incline', 'lean']:
         return 'tilt'


#import stoplist that removes stop words from
a document
stoplist = stopwords.words('english')

#import wordnet lemmatizer
wordnet_lemmatizer = WordNetLemmatizer()

#speech_recognition feature
r = sr.Recognizer()
with sr.Microphone() as source:
   print("Say");
   audio = r.listen(source)
   print("Time over, Thanks")
try:
   sentence = r.recognize_google(audio)
   tokens = preprocess(sentence)
   print (predict_intent(tokens))
except:
   pass
```

Conclusion

To develop and manage a production-ready model, work flow passes through the following stages:

1. Source and prepare the data.
2. Develop the model.
3. Train an ML model on your data:

    - Train model
    - Evaluate model accuracy
    - Tune hyper parameters

4. Deploy the trained model.

5. Send prediction requests to your model:

    - Online prediction
    - Batch prediction

6. Monitor the predictions on an ongoing basis.

7. Manage the models and model versions.

These stages are iterative if required to reevaluate and go back to a previous step at any point in the process.

Machine learning can automatically run on implicit patterns and extract valuable information while accounting for the inherent complexity of cement mixtures and their properties. The above state of Machine learning adoption in cement mixer is an eye opener for similar types of design and development in any industry activity.