Name - Nilesh Choudhary

College - Lovely Professional University.

Project - Financial Risk Prediction Using Ibm Auto Ai.

**Project Documentation**

1. Introduction :-

Automation and artificial intelligence (AI) are transforming businesses and will contribute to economic growth via contributions to productivity. They will also help address challenges in areas of healthcare, technology & other areas. At the same time, these technologies will transform the nature of work and the workplace itself. In this code pattern, we will focus on building state of the art systems for churning out predictions which can be used in different scenarios. We will try to predict fraudulent transactions which we know can reduce monetary loss and risk mitigation. The same approach can be used for predicting customer churn, demand and supply forecast and others. Building predictive models require time, effort and good knowledge of algorithms to create effective systems which can predict the outcome accurately. With that being said, IBM has introduced Auto AI which will automate all the tasks involved in building predictive models for different requirements. We will get to see how Auto AI can churn out great models quickly which will save time and effort and aid in faster decision making process.

2. Literature Survey:-

A business can lose a significant amount of [assets](https://www.accountingtools.com/articles/what-is-an-asset.html) due to [fraud](https://www.accountingtools.com/articles/what-is-fraud.html). At an extreme level, the effects of fraud can even shut down a company. Consequently, a business owner should make ongoing efforts to create an environment in which fraud is less likely to arise. There are a number of factors that make it more likely that fraud will occur or is occurring in a business. These fraud risk factors include:

## Nature of Items

* *Size and value*. If items that can be stolen are of high value in proportion to their size (such as diamonds), it is less risky to remove them from the premises. This is a particularly critical item if it is easy for [employees](https://www.accountingtools.com/articles/2017/5/4/definition-of-an-employee) to do so.
* *Ease of resale*. If there is a ready market for the resale of stolen goods (such as for most types of consumer electronics), this presents an increased temptation to engage in fraud.
* *Cash*. If there is a large amount of bills and coins on hand, or [cash](https://www.accountingtools.com/articles/2017/5/4/cash) in [bank accounts](https://www.accountingtools.com/articles/2017/5/15/bank-account-types), there is a very high risk of fraud. At a local level, a large balance in a [petty cash box](https://www.accountingtools.com/articles/petty-cash-box-usage-issues.html) presents a considerable temptation.

## Nature of Control Environment

* [*Separation of duties*](https://www.accountingtools.com/articles/what-is-separation-of-duties.html). The risk of fraud declines dramatically if multiple employees are involved in different phases of a [transaction](https://www.accountingtools.com/articles/2017/5/15/transaction), since fraud requires the [collusion](https://www.accountingtools.com/articles/collusion.html) of at least two people. Thus, poorly-defined job descriptions and approval processes present a clear opportunity for fraud.
* *Safeguards*. When assets are physically protected, they are much less likely to be stolen. This can involve fencing around the [inventory](https://www.accountingtools.com/articles/2017/5/13/inventory) storage area, a locked bin for maintenance supplies and tools, security guard stations, an employee badge system, and similar solutions.
* *Documentation*. When there is no physical or electronic record of a transaction, employees can be reasonably assured of not being caught, and so are more inclined to engage in fraud. This is also the case when there *is* documentation, but the records can be easily modified.
* *Time off*. When a business requires its employees to take the full amount of allocated time off, this keeps them from continuing to hide ongoing cases of fraud, and so is a natural deterrent.
* *Related party transactions*. When there are numerous transactions with [related parties](https://www.accountingtools.com/articles/2017/5/11/related-party), it is more likely that purchases and sales will be made at amounts that differ considerably from the [market price](https://www.accountingtools.com/articles/what-is-market-price.html).
* *Complexity*. When the nature of a company's business involves very complex transactions, and especially ones involving estimates, it is easier for employees to manipulate the results of these transactions to report better results than is really the case.
* *Dominance*. When a single individual is in a position to dominate the decisions of the management team, and especially when the [board of directors](https://www.accountingtools.com/articles/2017/5/10/board-of-directors) is weak, this individual is more likely to engage in unsuitable behavior.
* *Turnover*. When there is a high level of turnover among the management team and among employees in general, the institutional memory regarding how transactions are processed is weakened, resulting in less attention to controls.
* *Auditing*. When there is no [internal audit](https://www.accountingtools.com/articles/2017/9/27/internal-audit) function, it is unlikely that incorrect or inappropriate transactions will be spotted or corrected.

3. Theoretical Analysis:-

Using IBM AutoAI, you automate all the tasks involved in building predictive models for different requirements. You see how AutoAI generates great models quickly which save time and effort and aid in faster decision-making process. You create a model that from a data set that includes the age, sex, BMI, number-of-children, smoking preferences, region and charges to predict the health insurance premium cost that an individual pays.

When you have completed this code pattern, you understand how to:

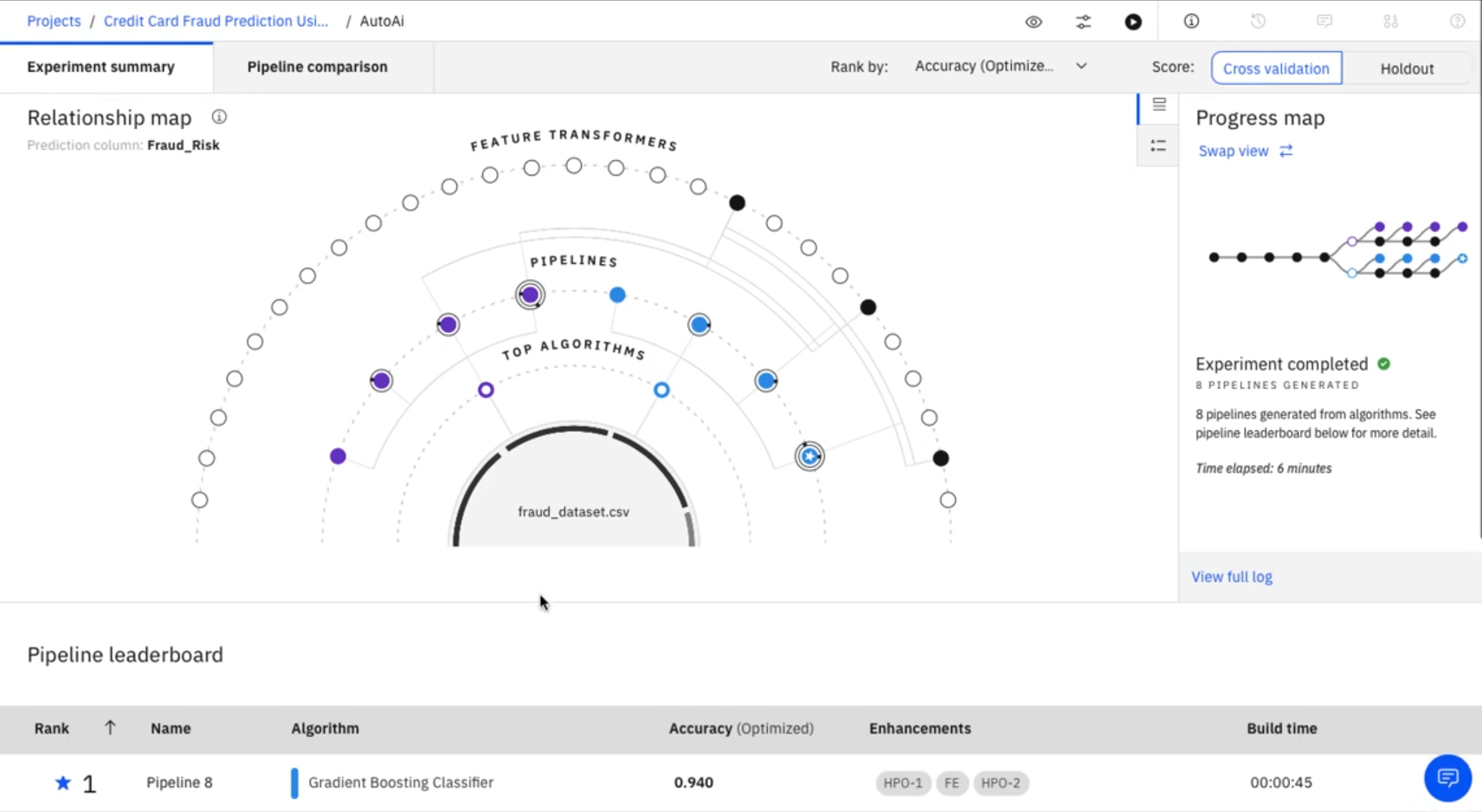
* Setup, quickly, the services on IBM Cloud for building the model.
* Ingest the data and initiate the AutoAI process.
* Build different models using AutoAI and evaluate the performance.
* Choose the best model and complete the deployment.
* Generate predictions using the deployed model by making REST calls.
* Compare the process of using AutoAI and building the model manually.
* Visualize the deployed model using a front-end application.

4. Experimental Investigations:-

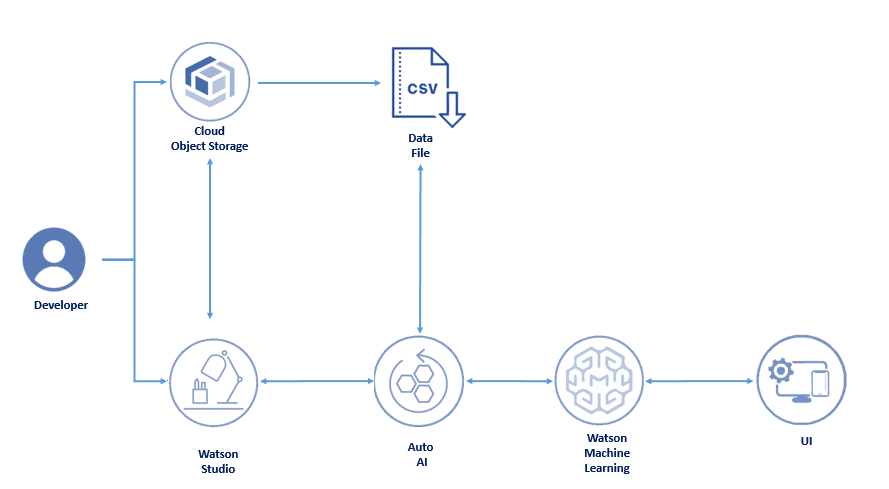
After associating data set with Ai model a experiment is set. This experiment is to creat different

pipelines with different levels of accuracy. The best set is considered the best pipeline and

furthur that is taken into consideration.



5. Flowchart :-



1- Creates an IBM Watson Studio Service on IBM Cloud.

2- Creates an IBM Cloud Object Storage Service and adds that to Watson Studio.

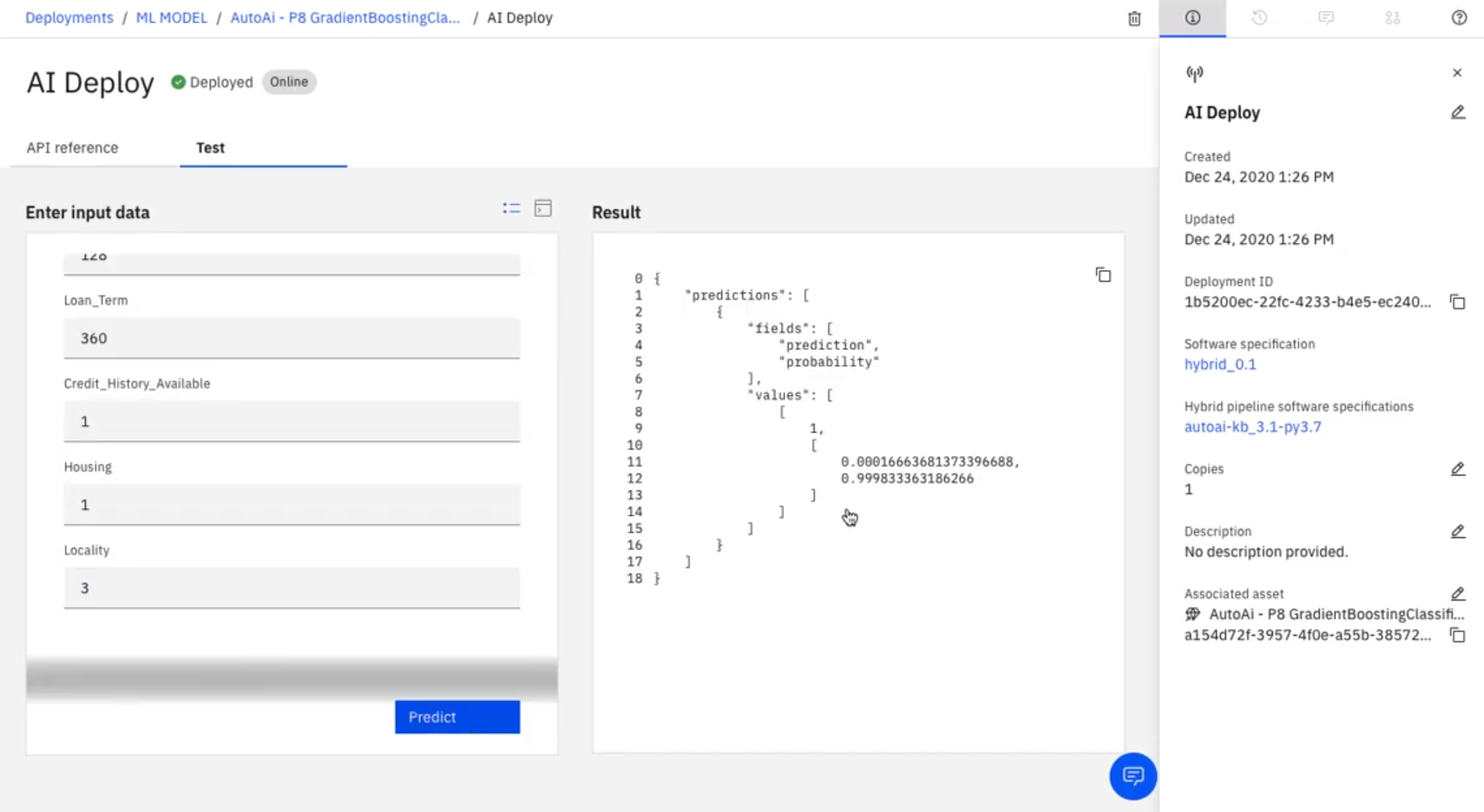
3- Uploads the insurance premium data file into Watson Studio.

4- Creates an AutoAI Experiment to predict insurance premium on Watson Studio

5- AutoAI uses Watson Machine Learning to create several models, and the user deploys the best performing model.

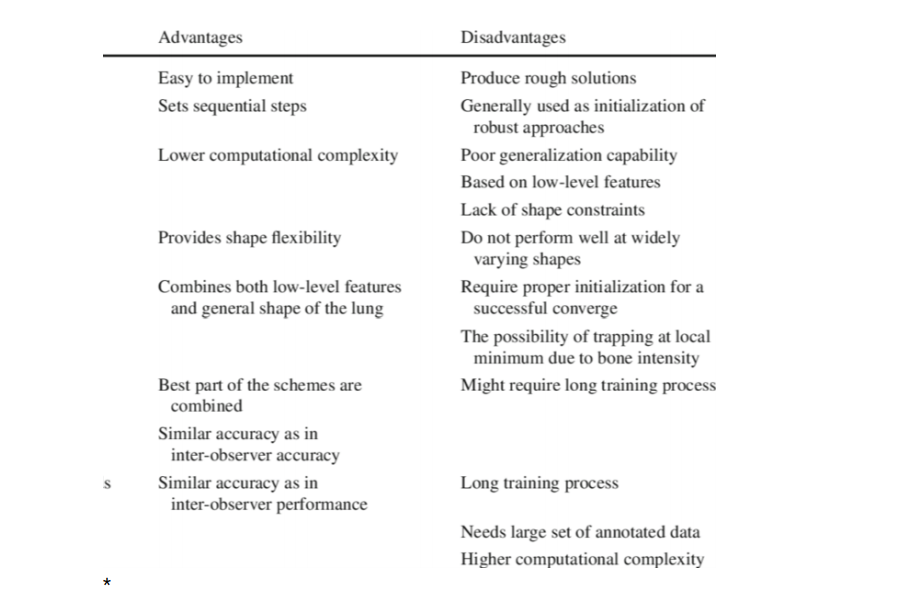
6- The user uses the Flask web-application to connect to the deployed model and predict an insurance charge

6. Result :-



The model shows how the finnacial risk is calculated using the given information

7. Advantages & Disadvantages: -



8. Applications :-

Each Individual can use it.

* *Level of dissatisfaction*. If the workforce is unhappy with the company, they will be more inclined to engage in fraud. Examples of such situations are when a [layoff](https://www.accountingtools.com/articles/2018/1/14/layoff) is imminent, benefits have been reduced, bonuses have been eliminated, promotions have been voided, and so forth.
* *Expectations*. When there is pressure from outside [investors](https://www.accountingtools.com/articles/2017/5/10/investor) to report certain financial results, or by management to meet certain performance targets (perhaps to earn bonuses), or to meet [balance sheet](https://www.accountingtools.com/articles/2017/5/17/the-balance-sheet) goals to qualify for [debt financing](https://www.accountingtools.com/articles/2017/5/6/debt-financing), there is a high risk of financial reporting fraud.
* *Guarantees*. When the owners or members of management have guaranteed company [debt](https://www.accountingtools.com/articles/2017/5/6/debt), there will be strong pressure to report certain financial results in order to avoid triggering the guarantees.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

9. Conclusion:-

Artificial Intelligence and Machine Learning are products of both science and myth. The idea that machines could think and perform tasks just as humans do is thousands of years old. The cognitive truths expressed in AI and Machine Learning systems are not new either. It may be better to view these technologies as the implementation of powerful and long-established cognitive principles through engineering.

We should accept that there is a tendency to approach all important innovations as a Rorschach test upon which we impose anxieties and hopes about what constitutes a good or happy world. But the potential of AI and machine intelligence for good does not lie exclusively, or even primarily, within its technologies. It lies mainly in its users. If we trust (in the main) how our societies are currently being run then we have no reason not to trust ourselves to do good with these technologies. And if we can suspend presentism and accept that ancient stories warning us not to play God with powerful technologies are instructive then we will likely free ourselves from unnecessary anxiety about their use.

10. Future Scope :-

Artificial Intelligence(AI) is the simulation of human intelligence by machines. In other words, it is the method by which machines demonstrate certain aspects of human intelligence like learning, reasoning and self- correction. Since its inception, AI has demonstrated unprecedented growth. Sophia the AI Robot, is the quintessential example of this. The future of Artificial intelligence is hazy. But going by the bounds of progress AI has been making, it is clear AI will permeate every sphere of our life. Listed below are the diverse ways in which AI can change in the future.

The scope of AI in science is the largest. Recently ‘Eve’ was in the news for discovering that an ingredient found commonly in toothpaste, is capable of curing Malaria. Here the subject in appreciation ‘Eve’ is not a human scientist, rather a Robot created by a team of scientists at the Universities of Manchester, Aberystwyth, and Cambridge.

The future application of AI in cybersecurity will ensure in curbing hackers. The incidence of cybercrime is an issue that has been escalating through the years. It costs enterprises in term of brand image as well as material cost. Credit card fraudery is one of the most prevalent cybercrimes. Despite there being detection techniques, they still prove to be ineffective in curbing hackers. AI can bring a remarkable change to this. Novel AI techniques like Recurrent Neural Networks can detect fraudery in initial stages itself. This fraud detection system will be able to scan thousands of transactions instantly and predict/ classify them into buckets. RNN can save a lot of time as it focuses on cases where there is a high probability for fraud.

Tech has advanced in terms of Emotional Quotient. Virtual assistants [Siri](https://www.apple.com/in/ios/siri/), Cortana & Alexa show how the extent to which AI comprehends human language. They are able to understand the meaning from context and make intelligent judgments. Back in 2015, a companion robot called, ‘Pepper’ went on sale. All the initial 1000 units were sold within a minute. Overall, considering all this, the possibility of emotional bots might become a reality in the future.

11. Bibliography:-

* IBM CLOUD.
* PYTHON.ORG
* MEDIUM.COM
* WATSON STUDIOS LEARNING.
* NODE RED LEARNING.
* SMARTINTERNZ.COM.
* YOUTUBE.COM.
* WIKIPEDIA.

12. Appendix:-

Source code-

#attempt import of autoai\_libs and install if missing

try:

import autoai\_libs

except Exception as e:

print('attempting to install missing autoai\_libs from pypi, this may take tens of seconds to complete.')

import subprocess

try:

# attempt to install missing autoai-libs from pypi

out = subprocess.check\_output('pip install autoai-libs', shell=True)

for line in out.splitlines():

print(line)

except Exception as e:

print(str(e))

try:

import autoai\_libs

except Exception as e:

print('attempting to install missing autoai\_libs from local filesystem, this may take tens of seconds to complete.')

import subprocess

# attempt to install missing autoai-libs from local filesystem

try:

out = subprocess.check\_output('pip install .', shell=True, cwd='software/autoai\_libs')

for line in out.splitlines():

print(line)

import autoai\_libs

except Exception as e:

print(str(e))

import sklearn

try:

import xgboost

except:

print('xgboost, if needed, will be installed and imported later')

try:

import lightgbm

except:

print('lightgbm, if needed, will be installed and imported later')

from sklearn.cluster import FeatureAgglomeration

import numpy

from numpy import inf, nan, dtype, mean

from autoai\_libs.sklearn.custom\_scorers import CustomScorers

from autoai\_libs.cognito.transforms.transform\_utils import TExtras, FC

from autoai\_libs.transformers.exportable import \*

from autoai\_libs.utils.exportable\_utils import \*

from sklearn.pipeline import Pipeline

known\_values\_list=[]

# compose a decorator to assist pipeline instantiation via import of modules and installation of packages

def decorator\_retries(func):

def install\_import\_retry(\*args, \*\*kwargs):

retries = 0

successful = False

failed\_retries = 0

while retries < 100 and failed\_retries < 10 and not successful:

retries += 1

failed\_retries += 1

try:

result = func(\*args, \*\*kwargs)

successful = True

except Exception as e:

estr = str(e)

if estr.startswith('name ') and estr.endswith(' is not defined'):

try:

import importlib

module\_name = estr.split("'")[1]

module = importlib.import\_module(module\_name)

globals().update({module\_name: module})

print('import successful for ' + module\_name)

failed\_retries -= 1

except Exception as import\_failure:

print('import of ' + module\_name + ' failed with: ' + str(import\_failure))

import subprocess

print('attempting pip install of ' + module\_name)

process = subprocess.Popen('pip install ' + module\_name, shell=True)

process.wait()

try:

print('re-attempting import of ' + module\_name)

module = importlib.import\_module(module\_name)

globals().update({module\_name: module})

print('import successful for ' + module\_name)

failed\_retries -= 1

except Exception as import\_or\_installation\_failure:

print('failure installing and/or importing ' + module\_name + ' error was: ' + str(

import\_or\_installation\_failure))

raise (ModuleNotFoundError('Missing package in environment for ' + module\_name +

'? Try import and/or pip install manually?'))

elif type(e) is AttributeError:

if 'module ' in estr and ' has no attribute ' in estr:

pieces = estr.split("'")

if len(pieces) == 5:

try:

import importlib

print('re-attempting import of ' + pieces[3] + ' from ' + pieces[1])

module = importlib.import\_module('.' + pieces[3], pieces[1])

failed\_retries -= 1

except:

print('failed attempt to import ' + pieces[3])

raise (e)

else:

raise (e)

else:

raise (e)

if successful:

print('Pipeline successfully instantiated')

else:

raise (ModuleNotFoundError(

'Remaining missing imports/packages in environment? Retry cell and/or try pip install manually?'))

return result

return install\_import\_retry

# Metadata used in retrieving data and computing metrics. Customize as necessary for your environment.

#data\_source='replace\_with\_path\_and\_csv\_filename'

target\_label\_name = \_input\_metadata['target\_label\_name']

learning\_type = \_input\_metadata['learning\_type']

optimization\_metric = \_input\_metadata['optimization\_metric']

random\_state = \_input\_metadata['random\_state']

cv\_num\_folds = \_input\_metadata['cv\_num\_folds']

holdout\_fraction = \_input\_metadata['holdout\_fraction']

if 'data\_provenance' in \_input\_metadata:

data\_provenance = \_input\_metadata['data\_provenance']

else:

data\_provenance = None

if 'pos\_label' in \_input\_metadata and learning\_type == 'classification':

pos\_label = \_input\_metadata['pos\_label']

else:

pos\_label = None

# @hidden\_cell

# The following code contains the credentials for a file in your IBM Cloud Object Storage.

# You might want to remove those credentials before you share your notebook.

credentials\_0 = {

}

# Read the data as a dataframe

import pandas as pd

csv\_encodings=['UTF-8','Latin-1'] # supplement list of encodings as necessary for your data

df = None

readable = None # if automatic detection fails, you can supply a filename here

# First, obtain a readable object

# IBM Cloud Object Storage data access

# Assumes COS credentials are in a dictionary named 'credentials\_0'

cos\_credentials = df = globals().get('credentials\_0')

if readable is None and cos\_credentials is not None:

print('accessing data via IBM Cloud Object Storage')

try:

import types

from botocore.client import Config

import ibm\_boto3

def \_\_iter\_\_(self): return 0

if 'SERVICE\_NAME' not in cos\_credentials: # in case of Studio-supplied credentials for a different dataset

cos\_credentials['SERVICE\_NAME'] = 's3'

client = ibm\_boto3.client(service\_name=cos\_credentials['SERVICE\_NAME'],

ibm\_api\_key\_id=cos\_credentials['IBM\_API\_KEY\_ID'],

ibm\_auth\_endpoint=cos\_credentials['IBM\_AUTH\_ENDPOINT'],

config=Config(signature\_version='oauth'),

endpoint\_url=cos\_credentials['ENDPOINT'])

try:

readable = client.get\_object(Bucket=cos\_credentials['BUCKET'],Key=cos\_credentials['FILE'])['Body']

# add missing \_\_iter\_\_ method, so pandas accepts readable as file-like object

if not hasattr(readable, "\_\_iter\_\_"): readable.\_\_iter\_\_ = types.MethodType( \_\_iter\_\_, readable )

except Exception as cos\_access\_exception:

print('unable to access data object in cloud object storage with credentials supplied')

except Exception as cos\_exception:

print('unable to create client for cloud object storage')

# IBM Cloud Pak for Data data access

project\_filename = globals().get('project\_filename')

if readable is None and 'credentials\_0' in globals() and 'ASSET\_ID' in credentials\_0:

project\_filename = credentials\_0['ASSET\_ID']

if project\_filename is not None:

print('attempting project\_lib access to ' + str(project\_filename))

try:

from project\_lib import Project

project = Project.access()

storage\_credentials = project.get\_storage\_metadata()

readable = project.get\_file(project\_filename)

except Exception as project\_exception:

print('unable to access data using the project\_lib interface and filename supplied')

# Use data\_provenance as filename if other access mechanisms are unsuccessful

if readable is None and type(data\_provenance) is str:

print('attempting to access local file using path and name ' + data\_provenance)

readable = data\_provenance

# Second, use pd.read\_csv to read object, iterating over list of csv\_encodings until successful

if readable is not None:

for encoding in csv\_encodings:

try:

df = pd.read\_csv(readable, encoding=encoding)

print('successfully loaded dataframe using encoding = ' + str(encoding))

break

except Exception as exception\_csv:

print('unable to read csv using encoding ' + str(encoding))

print('handled error was ' + str(exception\_csv))

if df is None:

print('unable to read file/object as a dataframe using supplied csv\_encodings ' + str(csv\_encodings))

print("Please use 'insert to code' on data panel to load dataframe.")

raise(ValueError('unable to read file/object as a dataframe using supplied csv\_encodings ' + str(csv\_encodings)))

if df is None:

print('Unable to access bucket/file in IBM Cloud Object Storage or asset in IBM Cloud Pak for Data with the parameters supplied.')

print('This is abnormal, but proceeding assuming the notebook user will supply a dataframe by other means.')

print("Please use 'insert to code' on data panel to load dataframe.")

# Drop rows whose target is not defined

target = target\_label\_name # your target name here

if learning\_type == 'regression':

df[target] = pd.to\_numeric(df[target], errors='coerce')

df.dropna('rows', how='any', subset=[target], inplace=True)

# extract X and y

df\_X = df.drop(columns=[target])

df\_y = df[target]

# Detach preprocessing pipeline (which needs to see all training data)

preprocessor\_index = -1

preprocessing\_steps = []

for i, step in enumerate(pipeline.steps):

preprocessing\_steps.append(step)

if step[0]=='preprocessor':

preprocessor\_index = i

break

if len(pipeline.steps) > preprocessor\_index+1 and pipeline.steps[preprocessor\_index + 1][0] == 'cognito':

preprocessor\_index += 1

preprocessing\_steps.append(pipeline.steps[preprocessor\_index])

if preprocessor\_index >= 0:

preprocessing\_pipeline = Pipeline(memory=pipeline.memory, steps=preprocessing\_steps)

pipeline = Pipeline(steps=pipeline.steps[preprocessor\_index+1:])# Preprocess X

# preprocessor should see all data for cross\_validate on the remaining steps to match autoai scores

known\_values\_list.clear() # known\_values\_list is filled in by the preprocessing\_pipeline if needed

preprocessing\_pipeline.fit(df\_X.values, df\_y.values)

X\_prep = preprocessing\_pipeline.transform(df\_X.values)

# determine learning\_type and perform holdout split (stratify conditionally)

if learning\_type is None:

# When the problem type is not available in the metadata, use the sklearn type\_of\_target to determine whether to stratify the holdout split

# Caution: This can mis-classify regression targets that can be expressed as integers as multiclass, in which case manually override the learning\_type

from sklearn.utils.multiclass import type\_of\_target

if type\_of\_target(df\_y.values) in ['multiclass', 'binary']:

learning\_type = 'classification'

else:

learning\_type = 'regression'

print('learning\_type determined by type\_of\_target as:',learning\_type)

else:

print('learning\_type specified as:',learning\_type)

from sklearn.model\_selection import train\_test\_split

if learning\_type == 'classification':

X, X\_holdout, y, y\_holdout = train\_test\_split(X\_prep, df\_y.values, test\_size=holdout\_fraction, random\_state=random\_state, stratify=df\_y.values)

else:

X, X\_holdout, y, y\_holdout = train\_test\_split(X\_prep, df\_y.values, test\_size=holdout\_fraction, random\_state=random\_state)

# create a function to produce a scorer for a given positive label

def make\_pos\_label\_scorer(scorer, pos\_label):

kwargs = {'pos\_label':pos\_label}

for prop in ['needs\_proba', 'needs\_threshold']:

if prop+'=True' in scorer.\_factory\_args():

kwargs[prop] = True

if scorer.\_sign == -1:

kwargs['greater\_is\_better'] = False

from sklearn.metrics import make\_scorer

scorer=make\_scorer(scorer.\_score\_func, \*\*kwargs)

return scorer

# fit the remainder of the pipeline on the training data

pipeline.fit(X,y)# predict on the holdout data

y\_pred = pipeline.predict(X\_holdout)# compute score for the optimization metric

# scorer may need pos\_label, but not all scorers take pos\_label parameter

from sklearn.metrics import get\_scorer

scorer = get\_scorer(optimization\_metric)

score = None

#score = scorer(pipeline, X\_holdout, y\_holdout) # this would suffice for simple cases

pos\_label = None # if you want to supply the pos\_label, specify it here

if pos\_label is None and 'pos\_label' in \_input\_metadata:

pos\_label=\_input\_metadata['pos\_label']

try:

score = scorer(pipeline, X\_holdout, y\_holdout)

except Exception as e1:

if pos\_label is None or str(pos\_label)=='':

print('You may have to provide a value for pos\_label in order for a score to be calculated.')

raise(e1)

else:

exception\_string=str(e1)

if 'pos\_label' in exception\_string:

try:

scorer = make\_pos\_label\_scorer(scorer, pos\_label=pos\_label)

score = scorer(pipeline, X\_holdout, y\_holdout)

print('Retry was successful with pos\_label supplied to scorer')

except Exception as e2:

print('Initial attempt to use scorer failed. Exception was:')

print(e1)

print('')

print('Retry with pos\_label failed. Exception was:')

print(e2)

else:

raise(e1)

if score is not None:

print(score)# cross\_validate pipeline using training data

from sklearn.model\_selection import cross\_validate

from sklearn.model\_selection import StratifiedKFold, KFold

if learning\_type == 'classification':

fold\_generator = StratifiedKFold(n\_splits=cv\_num\_folds, random\_state=random\_state)

else:

fold\_generator = KFold(n\_splits=cv\_num\_folds, random\_state=random\_state)

cv\_results = cross\_validate(pipeline, X, y, cv=fold\_generator, scoring={optimization\_metric:scorer}, return\_train\_score=True)

import numpy as np

np.mean(cv\_results['test\_' + optimization\_metric])cv\_results

Screenshot of UI:-

