Explaining_model_using_what_if_tool_for_ml_model

December 21, 2021

[1]:
```
# Importing all the required libraries

import pandas as pd
import xgboost as xgb
import numpy as np
import collections

# importing the what if tool to understand and explain the model that will be␣
 ↪built
import witwidget

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.utils import shuffle

# importing WhatIfTool visualisation tools
from witwidget.notebook.visualization import WitWidget, WitConfigBuilder
```

[2]:
```
# The mortgage dataset is fetched from "https://www.ffiec.gov/hmda/hmdaflat.
 ↪htm" originally
# Importing the dataset from cloud storage
!gsutil cp 'gs://mortgage_dataset_files/mortgage-small.csv' .
```

```
Copying gs://mortgage_dataset_files/mortgage-small.csv…
/ [1 files][330.8 MiB/330.8 MiB]
Operation completed over 1 objects/330.8 MiB.
```

[22]:
```
# Column Names of the dataset with their data types

COLUMN_NAMES = collections.OrderedDict({
 'as_of_year': np.int16,
 'agency_code': 'category',
 'loan_type': 'category',
 'property_type': 'category',
 'loan_purpose': 'category',
 'occupancy': np.int8,
 'loan_amt_thousands': np.float64,
```

```
 'preapproval': 'category',
 'county_code': np.float64,
 'applicant_income_thousands': np.float64,
 'purchaser_type': 'category',
 'hoepa_status': 'category',
 'lien_status': 'category',
 'population': np.float64,
 'ffiec_median_fam_income': np.float64,
 'tract_to_msa_income_pct': np.float64,
 'num_owner_occupied_units': np.float64,
 'num_1_to_4_family_units': np.float64,
 'approved': np.int8
})
```

[23]:
```
# ingesting the dataset
data = pd.read_csv(
 'mortgage-small.csv',
 index_col=False,
 dtype=COLUMN_NAMES
)

# Removing all the null values
data = data.dropna()

# shuffling the dataset
data = shuffle(data, random_state=2)
data.head()
```

[23]:

	as_of_year	agency_code	loan_type	property_type	loan_purpose
310650	2016	Consumer Financial Protection Bureau (CFPB)	Conventional (any loan other than FHA, VA, FSA…	One to four-family (other than manufactured ho…	Refinancing
630129	2016	Department of Housing and Urban Development (HUD)	Conventional (any loan other than FHA, VA, FSA…	One to four-family (other than manufactured ho…	Home purchase
715484	2016	Federal Deposit Insurance Corporation (FDIC)	Conventional (any loan other than FHA, VA, FSA…	One to four-family (other than manufactured ho…	Refinancing
887708	2016	Office of the Comptroller of the Currency (OCC)	Conventional (any loan other than FHA, VA, FSA…	One to four-family (other than manufactured ho…	Refinancing
719598	2016	National Credit Union Administration (NCUA)	Conventional (any loan other than FHA, VA, FSA…		

```
719598  One to four-family (other than manufactured ho…    Refinancing

        occupancy  loan_amt_thousands     preapproval  county_code  \
310650          1               110.0  Not applicable        119.0
630129          1               480.0  Not applicable         33.0
715484          2               240.0  Not applicable         59.0
887708          1                76.0  Not applicable         65.0
719598          1               100.0  Not applicable        127.0

        applicant_income_thousands  \
310650                        55.0
630129                       270.0
715484                        96.0
887708                        85.0
719598                        70.0

                                        purchaser_type      hoepa_status  \
310650                              Freddie Mac (FHLMC)  Not a HOEPA loan
630129  Loan was not originated or was not sold in cal…  Not a HOEPA loan
715484  Commercial bank, savings bank or savings assoc…  Not a HOEPA loan
887708  Loan was not originated or was not sold in cal…  Not a HOEPA loan
719598  Loan was not originated or was not sold in cal…  Not a HOEPA loan

                       lien_status  population  ffiec_median_fam_income  \
310650       Secured by a first lien      5930.0                  64100.0
630129       Secured by a first lien      4791.0                  90300.0
715484       Secured by a first lien      3439.0                 105700.0
887708  Secured by a subordinate lien      3952.0                  61300.0
719598       Secured by a first lien      2422.0                  46400.0

        tract_to_msa_income_pct  num_owner_occupied_units  \
310650                    98.81                    1305.0
630129                   144.06                    1420.0
715484                   104.62                     853.0
887708                    90.93                    1272.0
719598                    88.37                     650.0

        num_1_to_4_family_units  approved
310650                   1631.0         1
630129                   1450.0         0
715484                   1076.0         1
887708                   1666.0         1
719598                   1006.0         1
```

[24]:
```
# Class column is our target column. 0 stands for loan not approved and 1␣
↪stands for loan approved
print(data['approved'].value_counts())
```

```
# segregating the labels and the data
labels = data['approved'].values

# data without the target column
data = data.drop(columns=['approved'])
```

```
1    665389
0    334610
Name: approved, dtype: int64
```

[26]:
```
# changing all the categorical columns into dummy variables as XGboost only␣
↪takes in numerical values
dummy_columns = list(data.dtypes[data.dtypes == 'category'].index)
data = pd.get_dummies(data, columns=dummy_columns)

data.head()
```

[26]:

	as_of_year	occupancy	loan_amt_thousands	county_code	\
310650	2016	1	110.0	119.0	
630129	2016	1	480.0	33.0	
715484	2016	2	240.0	59.0	
887708	2016	1	76.0	65.0	
719598	2016	1	100.0	127.0	

	applicant_income_thousands	population	ffiec_median_fam_income	\
310650	55.0	5930.0	64100.0	
630129	270.0	4791.0	90300.0	
715484	96.0	3439.0	105700.0	
887708	85.0	3952.0	61300.0	
719598	70.0	2422.0	46400.0	

	tract_to_msa_income_pct	num_owner_occupied_units	\
310650	98.81	1305.0	
630129	144.06	1420.0	
715484	104.62	853.0	
887708	90.93	1272.0	
719598	88.37	650.0	

	num_1_to_4_family_units	…	\
310650	1631.0	…	
630129	1450.0	…	
715484	1076.0	…	
887708	1666.0	…	
719598	1006.0	…	

	purchaser_type_Life insurance company, credit union, mortgage bank, or

```
finance company  \
310650                                                  0
630129                                                  0
715484                                                  0
887708                                                  0
719598                                                  0

        purchaser_type_Loan was not originated or was not sold in calendar year
covered by register  \
310650                                                  0
630129                                                  1
715484                                                  0
887708                                                  1
719598                                                  1

        purchaser_type_Other type of purchaser  \
310650                                       0
630129                                       0
715484                                       0
887708                                       0
719598                                       0

        purchaser_type_Private securitization  hoepa_status_HOEPA loan  \
310650                                      0                         0
630129                                      0                         0
715484                                      0                         0
887708                                      0                         0
719598                                      0                         0

        hoepa_status_Not a HOEPA loan  \
310650                               1
630129                               1
715484                               1
887708                               1
719598                               1

        lien_status_Not applicable (purchased loans)  \
310650                                             0
630129                                             0
715484                                             0
887708                                             0
719598                                             0

        lien_status_Not secured by a lien  \
310650                                   0
630129                                   0
715484                                   0
```

```
887708                                    0
719598                                    0

        lien_status_Secured by a first lien  \
310650                                    1
630129                                    1
715484                                    1
887708                                    0
719598                                    1

        lien_status_Secured by a subordinate lien
310650                                          0
630129                                          0
715484                                          0
887708                                          1
719598                                          0

[5 rows x 44 columns]
```

[29]:
```
# splitting the training and testing data
x,y = data.values,labels
x_train,x_test,y_train,y_test = train_test_split(x,y)
```

[32]:
```
# intialising the classifier model
# chosing logistic regression as objective
model = xgb.XGBClassifier(
    objective='reg:logistic'
)
```

[33]:
```
# fitting the model on the training data
model.fit(x_train, y_train)
```

```
/opt/conda/lib/python3.7/site-packages/xgboost/sklearn.py:1224: UserWarning: The
use of label encoder in XGBClassifier is deprecated and will be removed in a
future release. To remove this warning, do the following: 1) Pass option
use_label_encoder=False when constructing XGBClassifier object; and 2) Encode
your labels (y) as integers starting with 0, i.e. 0, 1, 2, …, [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

[33]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=4,
              num_parallel_tree=1, objective='reg:logistic', predictor='auto',
              random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
```

```
                subsample=1, tree_method='exact', validate_parameters=1,
                verbosity=None)
```

We obtain the model with the above mentioned parameters

[34]:
```
# Predicting on our testing data
y_pred = model.predict(x_test)
acc = accuracy_score(y_test, y_pred.round())
print(acc, '\n')
```

```
0.874308
```

We obtained an accuracy of 87.4% which is more than the (approved/approved+not_approved) ratio which is 66%, this means our model has learnt somethinf from the data is not guseeing randomly.

[35]:
```
# saving the model in the memory
model.save_model('model.bst')
```

[36]:
```
# We are only taking first 2000 examples to understand our model with␣
 ↪"WhatIfTool"

num_wit_examples = 2000

# creating a hiorizontal stack of our test examples and reshaping
test_examples = np.hstack((x_test[:num_wit_examples],y_test[:num_wit_examples].
 ↪reshape(-1,1)))
```

[38]:
```
# configuring our WIT builder to analyse the model
config_builder = (WitConfigBuilder(test_examples.tolist(), data.columns.
 ↪tolist() + ['mortgage_status'])
  .set_custom_predict_fn(model.predict_proba)
  .set_target_feature('mortgage_status')
  .set_label_vocab(['denied', 'approved']))
# building the WIT Widget to visualise the model
WitWidget(config_builder, height=800)
```

```
<IPython.core.display.HTML object>
```

```
WitWidget(config={'model_type': 'classification', 'label_vocab': ['denied',␣
 ↪'approved'], 'feature_names': ['as…
```

[40]:
```
# initialising the variables for our GCP project
GCP_PROJECT = 'vertex-ai-projects'
MODEL_BUCKET = 'gs://xgb_mortgage_vertex_ai_projects'
MODEL_NAME = 'xgb_mortgage'
```

[41]:
```
!gsutil mb -l us-central1 $MODEL_BUCKET
```

```
Creating gs://xgb_mortgage_vertex_ai_projects/…
ServiceException: 409 A Cloud Storage bucket named
'xgb_mortgage_vertex_ai_projects' already exists. Try another name. Bucket names
must be globally unique across all Google Cloud projects, including those
outside of your organization.
```

[42]:
```
# copying our model from memroy to cloud storage
!gsutil cp ./model.bst $MODEL_BUCKET
```

```
Copying file://./model.bst [Content-Type=application/octet-stream]…
/ [1 files][290.6 KiB/290.6 KiB]
Operation completed over 1 objects/290.6 KiB.
```

[43]:
```
# uploading our model from cloud storage to models in VertexAI using container
 ↪to deply the model
!gcloud beta ai models upload \
--display-name=$MODEL_NAME \
--artifact-uri=$MODEL_BUCKET \
--container-image-uri=us-docker.pkg.dev/cloud-aiplatform/prediction/xgboost-cpu.
 ↪1-2:latest \
--region=us-central1
```

```
Using endpoint [https://us-central1-aiplatform.googleapis.com/]
Waiting for operation [7549294184029487104]…done.
```

We obtain the model endpoint which can be further used to deploy in the endpoint configurations

[44]:
```
# obtained after deploy of model on Vertex AI
MODEL_ID = "2917703637884993536"
```

Now we have to create an endpoint for outer world to access the model and predict with their data

[45]:
```
#Creating a model endpoint in the region us-central1
!gcloud beta ai endpoints create \
--display-name=xgb_mortgage_v1 \
--region=us-central1
```

```
Using endpoint [https://us-central1-aiplatform.googleapis.com/]
Waiting for operation [5741098933640232960]…done.
Created Vertex AI endpoint: projects/358157140210/locations/us-
central1/endpoints/5469137962325245952.
```

[46]:
```
# Endpoint id
ENDPOINT_ID = "5469137962325245952"
```

[47]:
```
# Deploying the endpoint on a compute engine machine (n1-standard-2)
!gcloud beta ai endpoints deploy-model $ENDPOINT_ID \
--region=us-central1 \
--model=$MODEL_ID \
```

```
--display-name=xgb_mortgage_v1 \
--machine-type=n1-standard-2 \
--traffic-split=0=100
```

```
Using endpoint [https://us-central1-aiplatform.googleapis.com/]
Waiting for operation [2302600618142859264]…done.
Deployed a model to the endpoint 5469137962325245952. Id of the deployed model:
6794075862074392576.
```

We can now access the deployed model for predictions

[48]:
```
#Testing with a sample data point
%%writefile predictions.json
{
  "instances": [
    [2016.0, 1.0, 346.0, 27.0, 211.0, 4530.0, 86700.0, 132.13, 1289.0, 1408.0,␣
 ↪0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.
 ↪0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.
 ↪0, 0.0, 1.0, 0.0]
  ]
}
```

```
Writing predictions.json
```

[49]:
```
# making a prediction with the sample data point
!gcloud beta ai endpoints predict $ENDPOINT_ID \
--json-request=predictions.json \
--region=us-central1
```

```
Using endpoint [https://us-central1-prediction-aiplatform.googleapis.com/]
[0.9999953508377075]
```

We get this prediction as approved since it gives 0.99 value in favour of “1”

[ ]:
```

```