Introduction

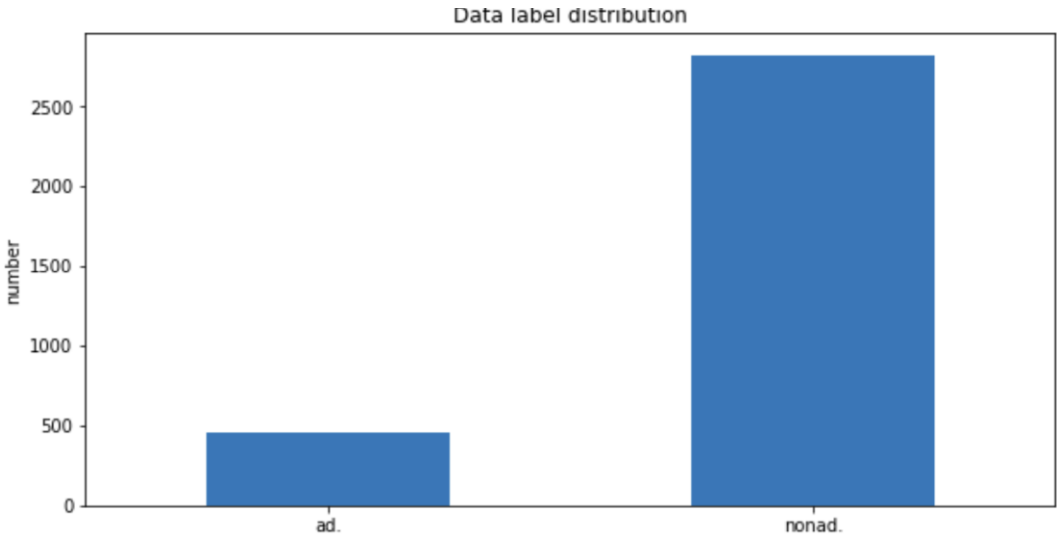
This project utilizes Azure ML Studio to train machine learning models to identify web ad elements. The dataset firstly goes through an exploration in Azure Notebook, then a pipeline is built in Designer to clean the dataset and train models for classifying ads. For each model among logistic regression and SVM, hyperparameters are tuned and performances are compared.

Dataset Overview

The dataset used include 3279 samples of web elements labeled as ad or non-ad. There are 1558 features, 3 of which are continuous (height, width and aspect ratio) and others are binary. The binary features indicate the presences of urls, as well as whether a keyword is included, in each sample. 28% of the samples have unknown continuous features denoted as '?'.

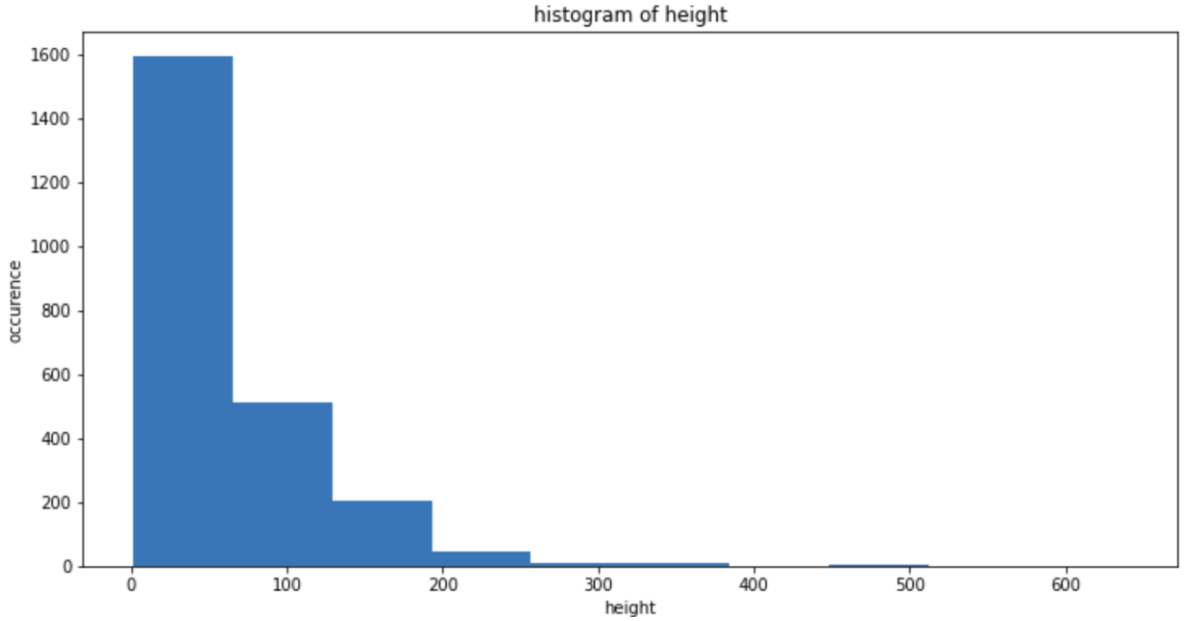
The purpose is to train a binary classification model to classify whether a web element sample is an ad with certain features in the dataset. The outcome model can help ad blocking apps to recognize ads and block them.

Overview of labels:



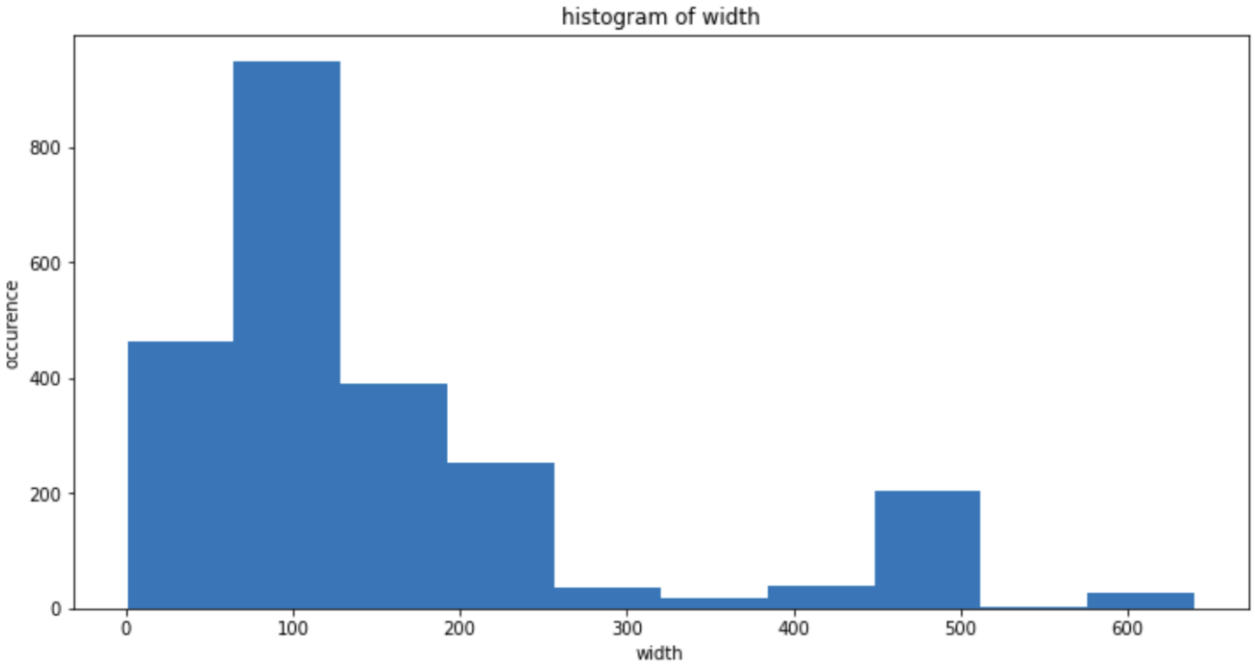
This dataset is seriously unbalanced. Thus the ad samples need to be duplicated 5 times to avoid a biased model.

Overview of height feature:



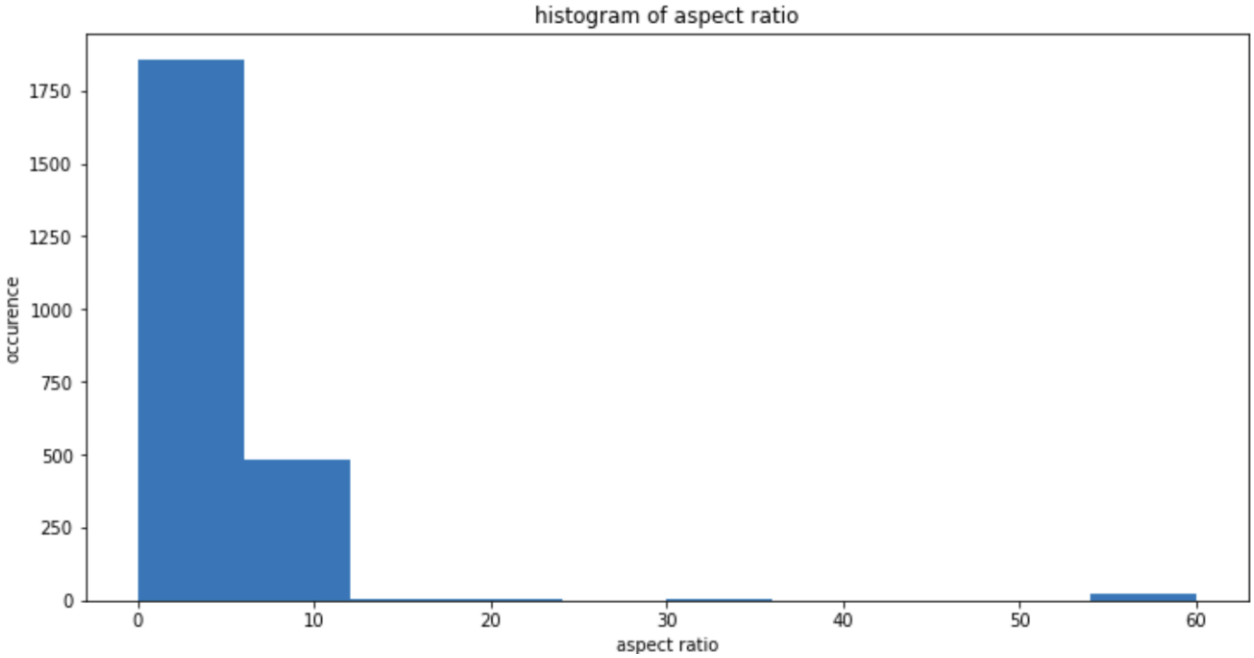
Most of them have height below 100. Unit is not given but estimated to be pixels.

Overview of width:



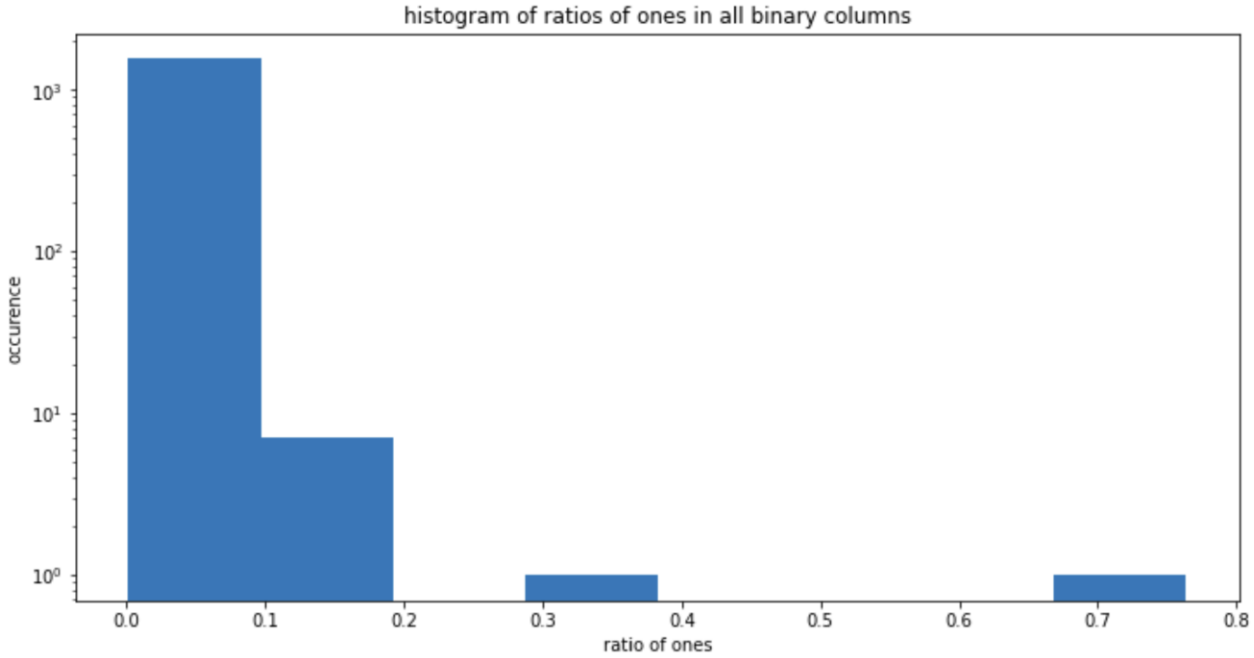
Most are within width of 250 with a mode of around 100, though there is another peak width of lower popularity of 450 to 500.

Overview of aspect ratio:



The majority are below 10 but some are in the range between 54 and 60.

Dataset sparsity of all columns:



Most of the binary features have portions of ones below 0.2, and the densest feature have portion of ones below 0.76. These very sparse features will likely be less helpful for model training as they are not representative and generalizable enough. Hence those features with too many zeros are excluded for model training.

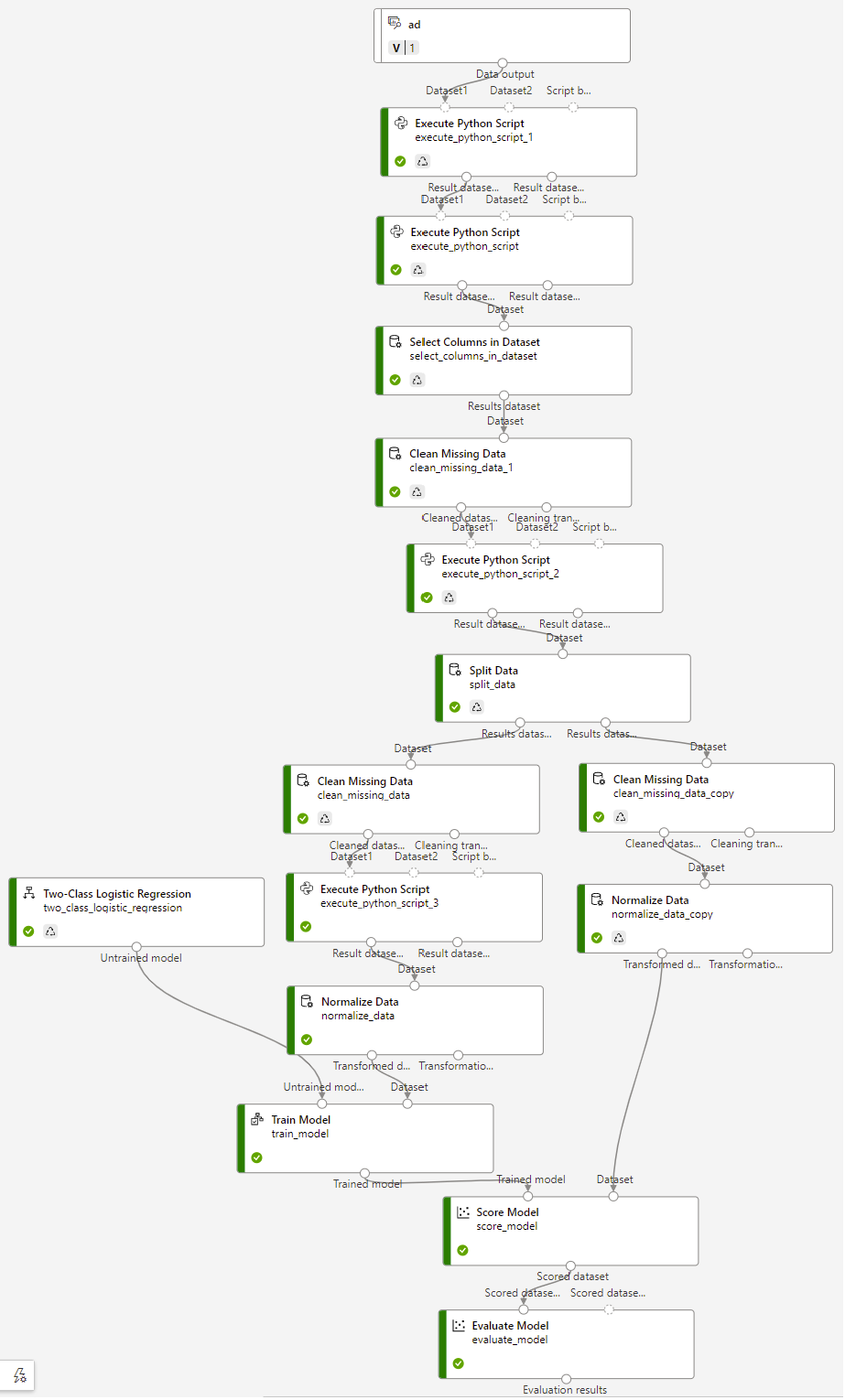
The first step to prepare training data is to select the 50 features with fewest zeros (hence the three continuous features are included) and discard the rest features. This will greatly improve efficiency and speed for later steps. Then the unknown values in each feature are replaced with median in the certain feature, for the sake of retaining its other features for training with minimal adverse effect by the missed data. Afterwards, all values except labels are converted to float, in order to keep datatype uniform and allows training to proceed. Later, the data is random split with a train-test ratio of 0.8:0.2. After splitting, the ad samples in the training dataset are duplicated for 5 times to make the two labels balanced, otherwise the model will be biased to predict non-ad more constantly.

In terms of machine learning models, one of the algorithms selected is two-class logistic regression. It is selected because it is suited for this binary classification problem. Also, logistic regression is a straight forward and efficient way to do such binary classification jobs with many features.

Another algorithm is two-class support vector machine. It is selected due to the robustness of SVM classifiers for non-linear and high dimensional classification jobs, as well as simpleness of hyperparameter tunning with only a lambda and number of iterations to tune.

Data cleaning and training

The data cleaning and training part are done in Azure ML Studio Designer. The pipeline is as shown in the picture below.



Code in execute_python_script_1, with the purpose to select 50 densest features:

```
import pandas as pd
import numpy as np
def azureml_main(dataframe1 = None, dataframe2 = None):

    # Execution logic goes here
    print(f'Input pandas.DataFrame #1: {dataframe1}')
    num0s=[]
    for i in range(dataframe1.shape[1]):
        colist=dataframe1.iloc[:,i]
        num0=0
        for val in colist:
            try:
                val=float(val)
            except:
                pass
            if val==0:
                num0+=1
            elif not type(val)==float:
                if '?' in val:
                    num0+=1
        num0s.append(num0)
    num0s
    num0s50=[]
    for i in range(51):
        minum=np.argmin(num0s)
        num0s50.append(minum)
        num0s[minum]=999999
    return dataframe1.iloc[:,num0s50],
```

Code in execute_python_script, with the purpose to substitute '?'s with None's, in order to enable the built-in Clean Missing Data block in Azure:

```
import pandas as pd
def azureml_main(dataframe1 = None, dataframe2 = None):
    print(f'Input pandas.DataFrame #1: {dataframe1}')
    try:
        dataframe1 = dataframe1.to_pandas_dataframe()
    except:
        pass
    a,b=dataframe1.shape
    for i in range(a):
        for j in range(b):
            try:
                if '?' in dataframe1.iloc[i,j]:
                    dataframe1.iloc[i,j]=None
            except:
                pass
    return dataframe1,
```

Code in execute_python_script_2, with the purpose to convert datatype of all values to float:

```
import pandas as pd
def azureml_main(dataframe1 = None, dataframe2 = None):
    print(f'Input pandas.DataFrame #1: {dataframe1}')
    a,b=dataframe1.shape
    for i in range(a):
        for j in range(b):
            try:
                dataframe1.iloc[i,j]=float(dataframe1.iloc[i,j])
            except:
                pass
    return dataframe1,
```

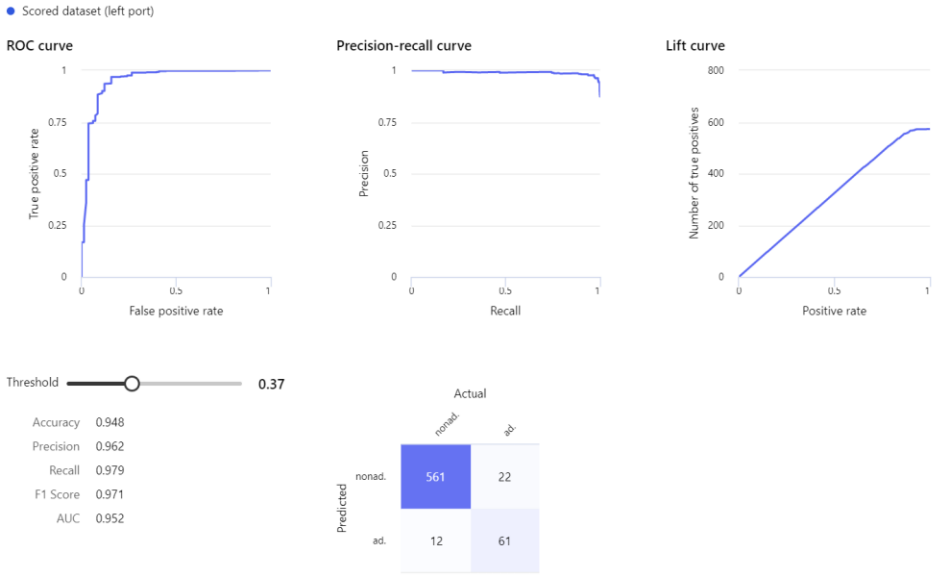
Code in execute_python_script_3, with the purpose to duplicate ad samples:

```
import pandas as pd
def azureml_main(dataframe1 = None, dataframe2 = None):
    print(f'Input pandas.DataFrame #1: {dataframe1}')
    adcnt=dataframe1.groupby('Column1559').count().iloc[0,0]
    dup=int(dataframe1.shape[0]/adcnt)-1
    df1=dataframe1.copy()
    appnum=0
    for i in range(0,len(dataframe1)):
        if dataframe1.loc[i,'Column1559']=='ad.':
            for j in range(dup):
                df1.loc[len(dataframe1)+appnum,:]=dataframe1.loc[i,:];
                appnum+=1;
    return df1,
```

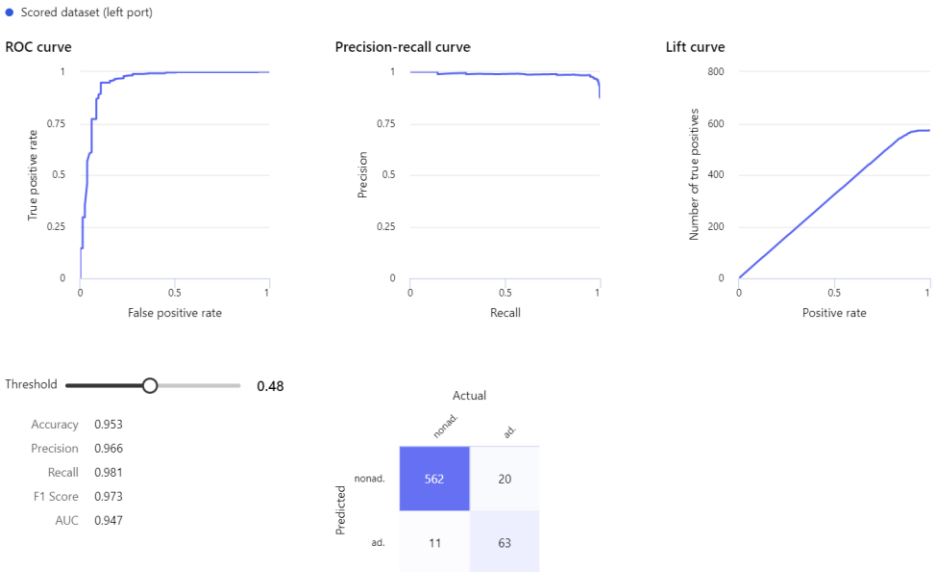
Results and discussion

The classification task cannot be simply judged with accuracy since false positives (non-ads classified as ads) and false negatives (ads classified as non-ads) have totally different effects in the reality. False negatives lead to more unblocked ads thus less functionality of the ad blocker, while false positives can mis-block useful information and cause serious results. The Azure ML designer platform allows a threshold adjustment for adjusting the bias to negative or positive predictions. In order to compare the performance of each model with each combination of hyperparameters, this threshold is adjusted to 2% false positive rate (11 in 573 samples) and then the false negative rate is compared.

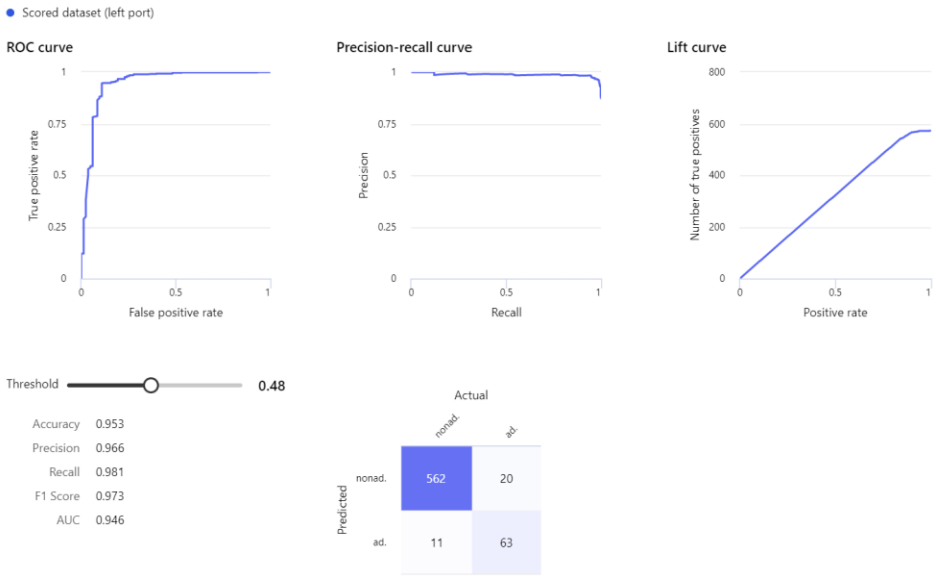
Two-class logistic regression,10 iterations:



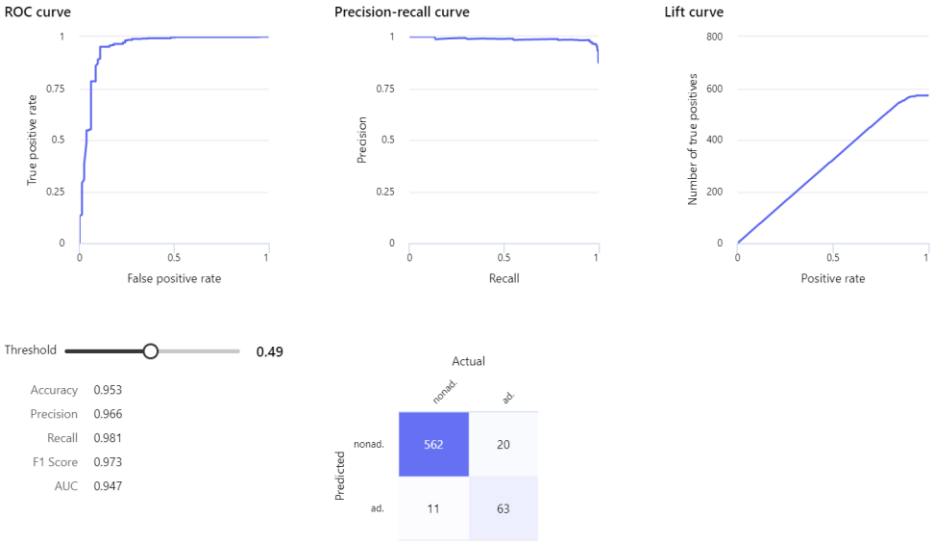
Two-class SVM, 10 iterations, lambda=0.001:



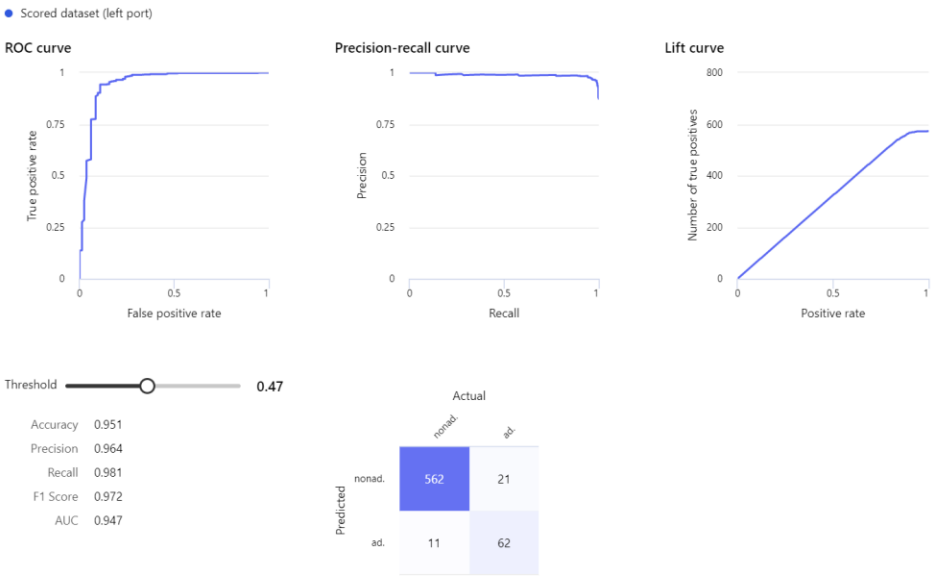
Two-class SVM, 18 iterations, lambda=0.001:



Two-class SVM, 12 iterations, lambda=0.01:



Two-class SVM, 12 iterations, lambda=0.1:



Overall, The two-class SVM with 10 iterations and lambda=0.001 has the best outcome by comparing the confusion matrix, then AUC, then the ROC curve. With a false positive rate of 0.0192, it gives a true positive rate of 0.759. In other words, over 3/4 of the ads can be detected and blocked, which is a decent performance.

Conclusion

This project conducted an exploration on a dataset about the features of ad and non-ad web elements, and built a pipeline to clean the data and train machine learning models to predict whether a web element is an ad. By limiting the false positive rate to 0.02, the best SVM classifier could detect over 3/4 of the ads. This model can be employed on a web ad blocking utility.