```
from google.colab import drive
drive.mount('/content/drive')

     Mounted at /content/drive
```

```
import os
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import warnings
warnings.filterwarnings("ignore")
import tensorflow as tf
from sklearn import model_selection
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

from __future__ import absolute_import, division, print_function, unicode_literals
from IPython.display import clear_output
from six.moves import urllib

import tensorflow.compat.v2.feature_column as fc

import tensorflow as tf
os.getcwd()
os.listdir('.')
print(os.getcwd())
print(os.chdir('/content/drive/MyDrive/Proj_colab'))


     /content
     None
```

```
# %% Read the original data and drop the columns
#originalD = pd.read_csv('data/Original_data.csv', low_memory=False)
originalD = pd.read_csv('data/Original_data1.csv', low_memory=False)
original_F = originalD.drop(['birthyr','faminc','employ','marstat','child18','pid3','pid7'
original_F
```

	id	gender	race	educ	region
0	371823339	Male	White	High School Graduate	Midwest
1	398212310	Male	White	High School Graduate	South
2	392933925	Male	White	No High School Degree	NorthEast
3	372445135	Male	White	High School Graduate	Midwest
4	392602384	Male	White	High School Graduate	South
...	...	...	...	...	...
4995	287972460	Female	Mixed Race	High School Graduate	Midwest
4996	137306469	Female	Mixed Race	High School Graduate	South

```
# %% Read the breached  data and drop the columns
breachD = pd.read_csv('data/breached_data.csv', low_memory=False)
breach_F = breachD.drop(['Title','Domain','Name','BreachDate','AddedDate','ModifiedDate','
breach_F.loc[:,'Breached'] ='1'
breach_F["Breached"] = breach_F["Breached"].astype(object).astype(int)
breach_F
```

	id	Breached
0	135664815	1
1	355286483	1
2	355286483	1
3	355286483	1
4	339141795	1
...	...	...
14974	131884325	1
14975	131884325	1
14976	131884325	1
14977	131884325	1
14978	131884325	1

14979 rows × 2 columns

```
breach_F1 = breach_F.drop_duplicates(subset =["id"] )
breach_F1["Breached"].replace({1: 0},inplace = True)
#df["column1"].replace({"a": "x", "b": "y"}, inplace=True)
#breach_F = breach_
#breach_F.loc[:,'Breached'] ='1'
breach_F1
```

	id	Breached
0	135664815	0
1	355286483	0
4	339141795	0
5	341961164	0
6	374206867	0
...	...	...
14960	137327203	0
14963	334328189	0
14967	151192859	0
14973	152094711	0
14974	131884325	0

```
df3 = pd.merge(breach_F, breach_F1, how='outer')
df3
```

	id	Breached
0	135664815	1
1	355286483	1
2	355286483	1
3	355286483	1
4	339141795	1
...	...	...
19116	137327203	0
19117	334328189	0
19118	151192859	0
19119	152094711	0
19120	131884325	0

19121 rows × 2 columns

```
# %% Merge the two files
fin_dat = pd.merge(original_F, df3, on='id', how='inner')
print("Number of rows in the final dataset: ", fin_dat.shape[0])
fin_dat.head(5)
```

```
Number of rows in the final dataset:  19121
```

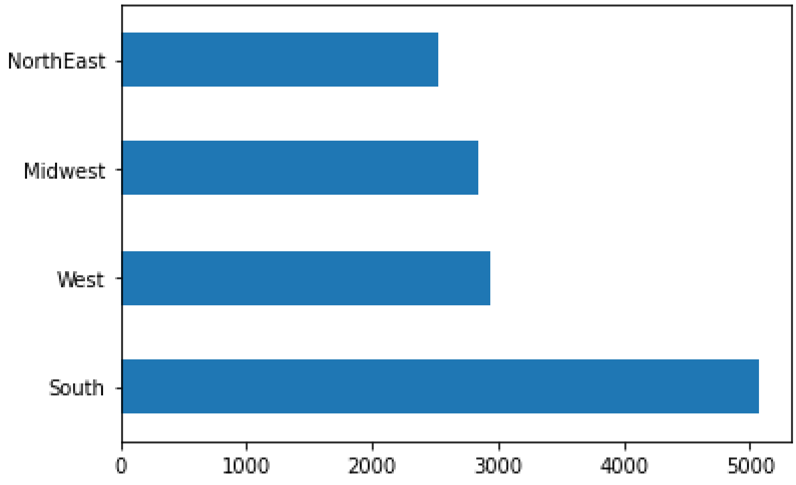
	id	gender	race	educ	region	Breached
0	371823339	Male	White	High School Graduate	Midwest	1
1	371823339	Male	White	High School Graduate	Midwest	1
2	371823339	Male	White	High School Graduate	Midwest	0
3	392933925	Male	White	No High School Degree	NorthEast	1

```
#input
x=fin_dat.drop('Breached',axis=1)
y= fin_dat.Breached
#splitting
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
#printing shapes of testing and training sets :
print("shape of original dataset :", fin_dat.shape)
print("shape of input - training set", x_train.shape)
print("shape of output - training set", y_train.shape)
print("shape of input - testing set", x_test.shape)
print("shape of output - testing set", y_test.shape)
```

```
shape of original dataset : (19121, 6)
shape of input - training set (13384, 5)
shape of output - training set (13384,)
shape of input - testing set (5737, 5)
shape of output - testing set (5737,)
```

```
x_train['region'].value_counts().plot(kind='barh')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe57e954550>
```

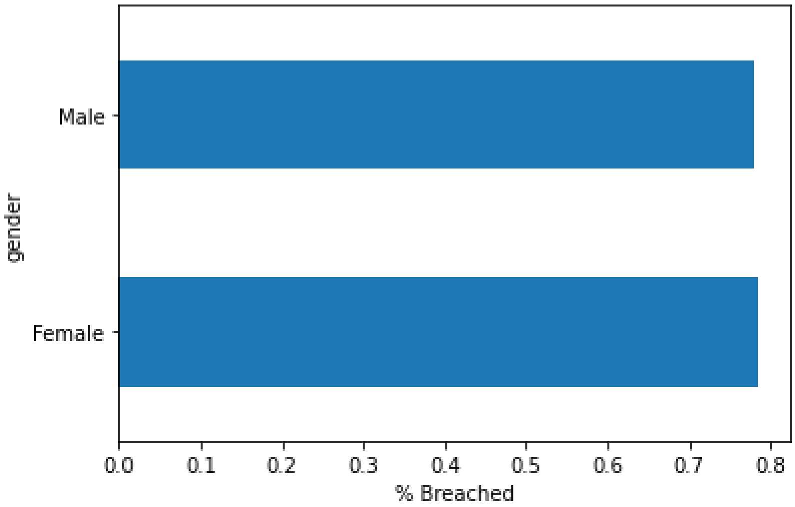


```
fin_dat.gender.value_counts().plot(kind='barh')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe57e946898>
```

```
pd.concat([x_train,y_train],axis=1).groupby('gender').Breached.mean().plot(kind ='barh').s
```

```
Text(0.5, 0, '% Breached')
```



```
CATEGORICAL_COLUMNS = ['gender', 'race', 'educ', 'region']
#NUMERIC_COLUMNS = ['age', 'fare']

feature_columns = []
for feature_name in CATEGORICAL_COLUMNS:
  vocabulary = x_train[feature_name].unique()  # gets a list of all unique values from giv
  feature_columns.append(tf.feature_column.categorical_column_with_vocabulary_list(feature_

#for feature_name in NUMERIC_COLUMNS:
 # feature_columns.append(tf.feature_column.numeric_column(feature_name, dtype=tf.float32)

print(feature_columns)
```

```
[VocabularyListCategoricalColumn(key='gender', vocabulary_list=('Male', 'Female'), dt
```

Double-click (or enter) to edit

```
def make_input_fn(data_df, label_df, num_epochs=10, shuffle=True, batch_size=32):
  def input_function():  # inner function, this will be returned
    ds = tf.data.Dataset.from_tensor_slices((dict(data_df), label_df))  # create tf.data.D
    if shuffle:
      ds = ds.shuffle(1000)  # randomize order of data
    ds = ds.batch(batch_size).repeat(num_epochs)  # split dataset into batches of 32 and r
    return ds  # return a batch of the dataset
  return input_function  # return a function object for use

train_input_fn = make_input_fn(x_train, y_train)  # here we will call the input_function t
eval_input_fn = make_input_fn(x_test,y_test , num_epochs=1, shuffle=False)
```

```
linear_est = tf.estimator.LinearClassifier(feature_columns=feature_columns)
# We create a linear estimtor by passing the feature columns we created earlier
```

```
INFO:tensorflow:Using default config.
WARNING:tensorflow:Using temporary folder as model directory: /tmp/tmpsvgr8ovs
INFO:tensorflow:Using config: {'_model_dir': '/tmp/tmpsvgr8ovs', '_tf_random_seed': N
graph_options {
  rewrite_options {
    meta_optimizer_iterations: ONE
  }
}
, '_keep_checkpoint_max': 5, '_keep_checkpoint_every_n_hours': 10000, '_log_step_cour
```

```
linear_est.train(train_input_fn)  # train
result = linear_est.evaluate(eval_input_fn)  # get model metrics/stats by testing on tetsi

clear_output()  # clears consoke output
print(result['accuracy'])  # the result variable is simply a dict of stats about our model
```

```
0.7856022
```

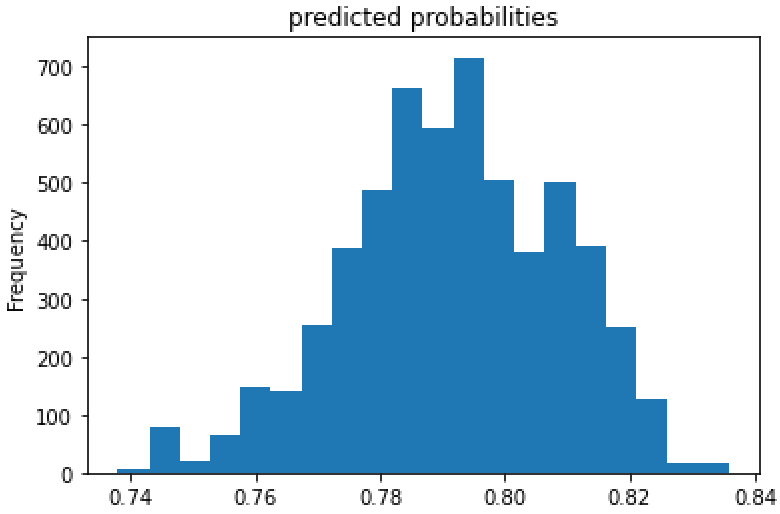
```
result = list(linear_est.predict(eval_input_fn))
print(x_train.loc[0])
print(result[1]['probabilities'])
```

```
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from /tmp/tmpsvgr8ovs/model.ckpt-4190
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
id                   371823339
gender                    Male
race                     White
educ      High School Graduate
region                 Midwest
Name: 0, dtype: object
[0.20341013 0.79658985]
```

```
pred_dicts = list(linear_est.predict(eval_input_fn))
probs = pd.Series([pred['probabilities'][1] for pred in pred_dicts])

probs.plot(kind='hist', bins=20, title='predicted probabilities')
```

```
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from /tmp/tmpsvgr8ovs/model.ckpt-4190
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
<matplotlib.axes._subplots.AxesSubplot at 0x7fe5763f2518>
```


predicted probabilities
Frequency