Decision Tree, Naive Bayes and Neural Network

November 13, 2021

[5]:
```
import pandas as pd
import matplotlib.pyplot as plt
from collections import Counter
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB

from sklearn.metrics import roc_curve, auc
from sklearn import metrics
from sklearn.neural_network import MLPClassifier
```

[8]:
```
Prudential_train = pd.read_csv("https://raw.githubusercontent.com/crisajose/
↪CIND-820-Big-Data-Analytics-Project/main/train.csv")
```

[9]:
```
Prudential_train.head()
```

[9]:

	Id	Product_Info_1	Product_Info_2	Product_Info_3	Product_Info_4	\
0	2	1	D3	10	0.076923	
1	5	1	A1	26	0.076923	
2	6	1	E1	26	0.076923	
3	7	1	D4	10	0.487179	
4	8	1	D2	26	0.230769	

	Product_Info_5	Product_Info_6	Product_Info_7	Ins_Age	Ht	…	\
0	2	1	1	0.641791	0.581818	…	
1	2	3	1	0.059701	0.600000	…	
2	2	3	1	0.029851	0.745455	…	
3	2	3	1	0.164179	0.672727	…	
4	2	3	1	0.417910	0.654545	…	

	Medical_Keyword_40	Medical_Keyword_41	Medical_Keyword_42	\
0	0	0	0	
1	0	0	0	
2	0	0	0	
3	0	0	0	

```
4                    0                    0                    0

   Medical_Keyword_43  Medical_Keyword_44  Medical_Keyword_45  \
0                   0                   0                   0
1                   0                   0                   0
2                   0                   0                   0
3                   0                   0                   0
4                   0                   0                   0

   Medical_Keyword_46  Medical_Keyword_47  Medical_Keyword_48  Response
0                   0                   0                   0         8
1                   0                   0                   0         4
2                   0                   0                   0         8
3                   0                   0                   0         8
4                   0                   0                   0         8

[5 rows x 128 columns]
```

[10]:
```
CATEGORICAL_COLUMNS = ["Product_Info_1", "Product_Info_2", "Product_Info_3",␣
↪"Product_Info_5", "Product_Info_6",\
                       "Product_Info_7", "Employment_Info_2",␣
↪"Employment_Info_3", "Employment_Info_5", "InsuredInfo_1",\
                       "InsuredInfo_2", "InsuredInfo_3", "InsuredInfo_4",␣
↪"InsuredInfo_5", "InsuredInfo_6", "InsuredInfo_7",\
                       "Insurance_History_1", "Insurance_History_2",␣
↪"Insurance_History_3", "Insurance_History_4", "Insurance_History_7",\
                       "Insurance_History_8", "Insurance_History_9",␣
↪"Family_Hist_1", "Medical_History_2", "Medical_History_3",\
                       "Medical_History_4", "Medical_History_5",␣
↪"Medical_History_6", "Medical_History_7", "Medical_History_8",\
                       "Medical_History_9", "Medical_History_11",␣
↪"Medical_History_12", "Medical_History_13", "Medical_History_14",\
                       "Medical_History_16", "Medical_History_17",␣
↪"Medical_History_18", "Medical_History_19", "Medical_History_20",\
                       "Medical_History_21", "Medical_History_22",␣
↪"Medical_History_23", "Medical_History_25", "Medical_History_26",\
                       "Medical_History_27", "Medical_History_28",␣
↪"Medical_History_29", "Medical_History_30", "Medical_History_31",\
                       "Medical_History_33", "Medical_History_34",␣
↪"Medical_History_35", "Medical_History_36", "Medical_History_37",\
                       "Medical_History_38", "Medical_History_39",␣
↪"Medical_History_40", "Medical_History_41"]
CONTINUOUS_COLUMNS = ["Product_Info_4", "Ins_Age", "Ht", "Wt", "BMI",
                      "Employment_Info_1", "Employment_Info_4",␣
↪"Employment_Info_6",
```

```
                    "Insurance_History_5", "Family_Hist_2", "Family_Hist_3",␣
 ↪"Family_Hist_4", "Family_Hist_5"]
DISCRETE_COLUMNS = ["Medical_History_1", "Medical_History_10",␣
 ↪"Medical_History_15", "Medical_History_24", "Medical_History_32"]
DUMMY_COLUMNS = ["Medical_Keyword_{}".format(i) for i in range(1, 48)]
```

[11]:
```
categorical_data = Prudential_train[CATEGORICAL_COLUMNS]
```

[12]:
```
continuous_data = Prudential_train[CONTINUOUS_COLUMNS]
```

[13]:
```
discrete_data = Prudential_train[DISCRETE_COLUMNS]
```

[14]:
```
dummy_data = Prudential_train[DUMMY_COLUMNS]
```

1 Variable Types

[16]:
```
Prudential_train.dtypes
```

[16]:
```
Id                      int64
Product_Info_1          int64
Product_Info_2         object
Product_Info_3          int64
Product_Info_4        float64
                        …
Medical_Keyword_45      int64
Medical_Keyword_46      int64
Medical_Keyword_47      int64
Medical_Keyword_48      int64
Response                int64
Length: 128, dtype: object
```

2 NULL values

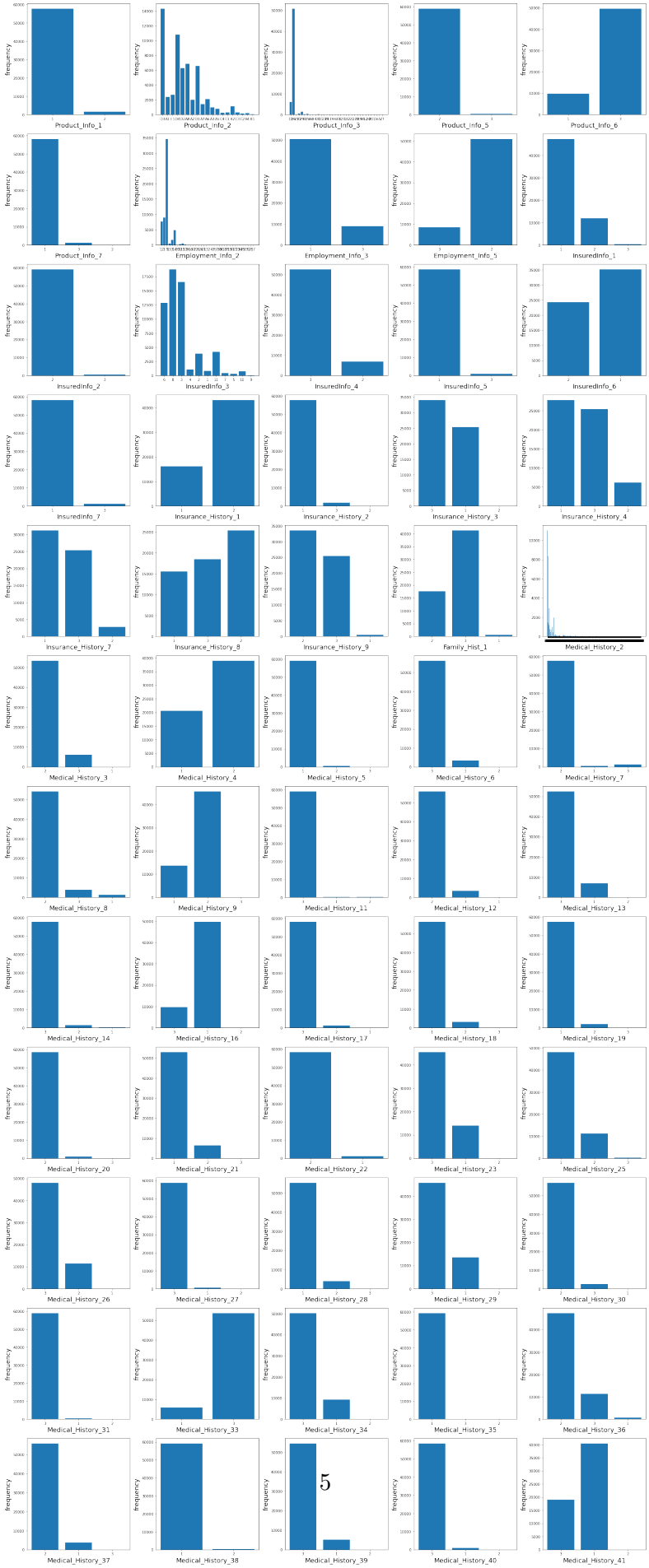
[17]:
```
Prudential_NULL = Prudential_train.isnull().any()
Prudential_NULL =[col for col in Prudential_train.columns if␣
 ↪Prudential_train[col].isnull().any()]
Prudential_NULL
```

[17]:
```
['Employment_Info_1',
 'Employment_Info_4',
 'Employment_Info_6',
 'Insurance_History_5',
 'Family_Hist_2',
```

```
 'Family_Hist_3',
 'Family_Hist_4',
 'Family_Hist_5',
 'Medical_History_1',
 'Medical_History_10',
 'Medical_History_15',
 'Medical_History_24',
 'Medical_History_32']
```

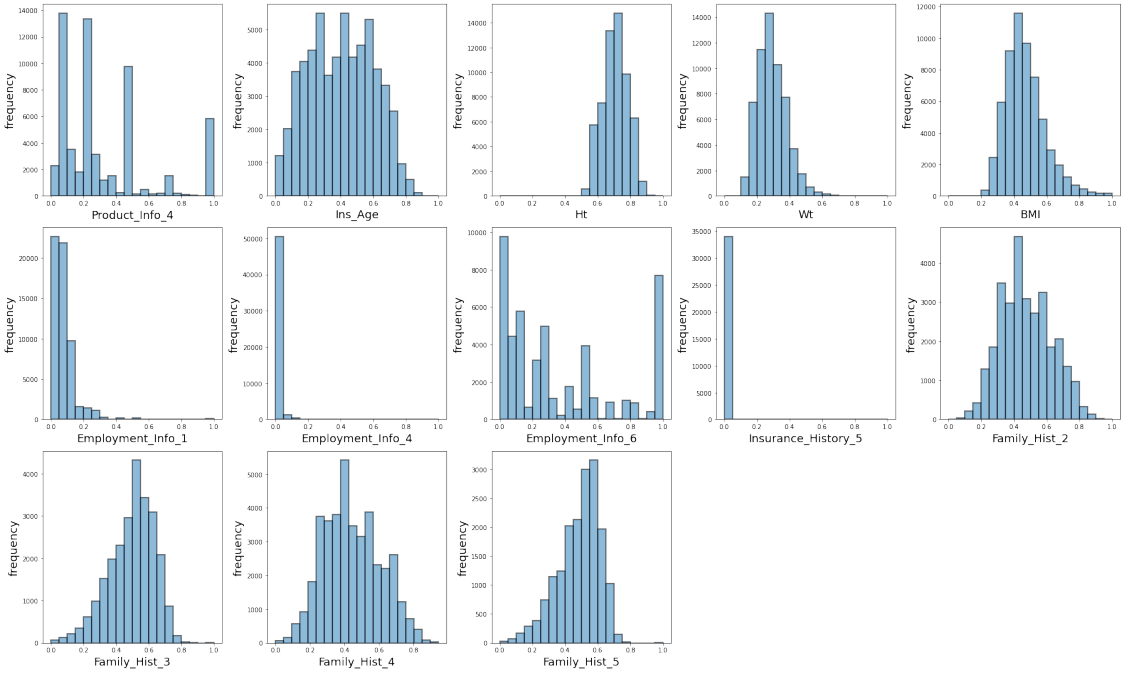
3 Categorical Variable - Plot

[18]:
```
def plot_categoricals(data):
    ncols = len(data.columns)
    fig = plt.figure(figsize=(5 * 5, 5 * (ncols // 5 + 1)))
    for i, col in enumerate(data.columns):
        cnt = Counter(data[col])
        keys = list(cnt.keys())
        vals = list(cnt.values())
        plt.subplot(ncols // 5 + 1, 5, i + 1)
        plt.bar(range(len(keys)), vals, align="center")
        plt.xticks(range(len(keys)), keys)
        plt.xlabel(col, fontsize=18)
        plt.ylabel("frequency", fontsize=18)
    fig.tight_layout()
    plt.show()

plot_categoricals(categorical_data)
```


frequency
Product_Info_1
Product_Info_2
Product_Info_3
Product_Info_5
Product_Info_6
Product_Info_7
Employment_Info_2
Employment_Info_3
Employment_Info_5
InsuredInfo_1
InsuredInfo_2
InsuredInfo_3
InsuredInfo_4
InsuredInfo_5
InsuredInfo_6
InsuredInfo_7
Insurance_History_1
Insurance_History_2
Insurance_History_3
Insurance_History_4
Insurance_History_7
Insurance_History_8
Insurance_History_9
Family_Hist_1
Medical_History_2
Medical_History_3
Medical_History_4
Medical_History_5
Medical_History_6
Medical_History_7
Medical_History_8
Medical_History_9
Medical_History_11
Medical_History_12
Medical_History_13
Medical_History_14
Medical_History_16
Medical_History_17
Medical_History_18
Medical_History_19
Medical_History_20
Medical_History_21
Medical_History_22
Medical_History_23
Medical_History_25
Medical_History_26
Medical_History_27
Medical_History_28
Medical_History_29
Medical_History_30
Medical_History_31
Medical_History_33
Medical_History_34
Medical_History_35
Medical_History_36
Medical_History_37
Medical_History_38
Medical_History_39
Medical_History_40
Medical_History_41

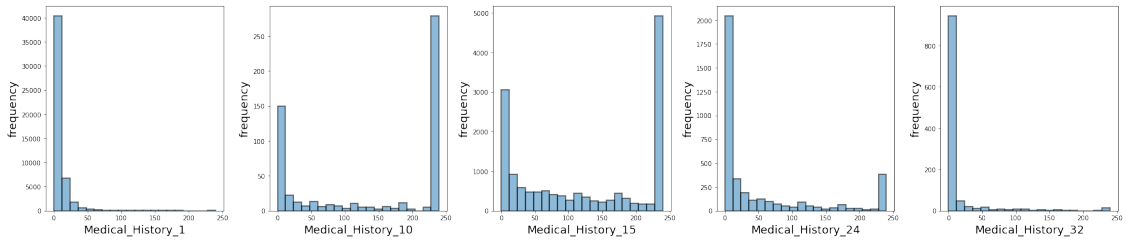
4 Continuous Variable - Plot

[19]:
```
def plot_histgrams(data):
    ncols = len(data.columns)
    fig = plt.figure(figsize=(5 * 5, 5 * (ncols // 5 + 1)))
    for i, col in enumerate(data.columns):
        X = data[col].dropna()
        plt.subplot(ncols // 5 + 1, 5, i + 1)
        plt.hist(X, bins=20, alpha=0.5, \
                 edgecolor="black", linewidth=2.0)
        plt.xlabel(col, fontsize=18)
        plt.ylabel("frequency", fontsize=18)
    fig.tight_layout()
    plt.show()
plot_histgrams(continuous_data)
```



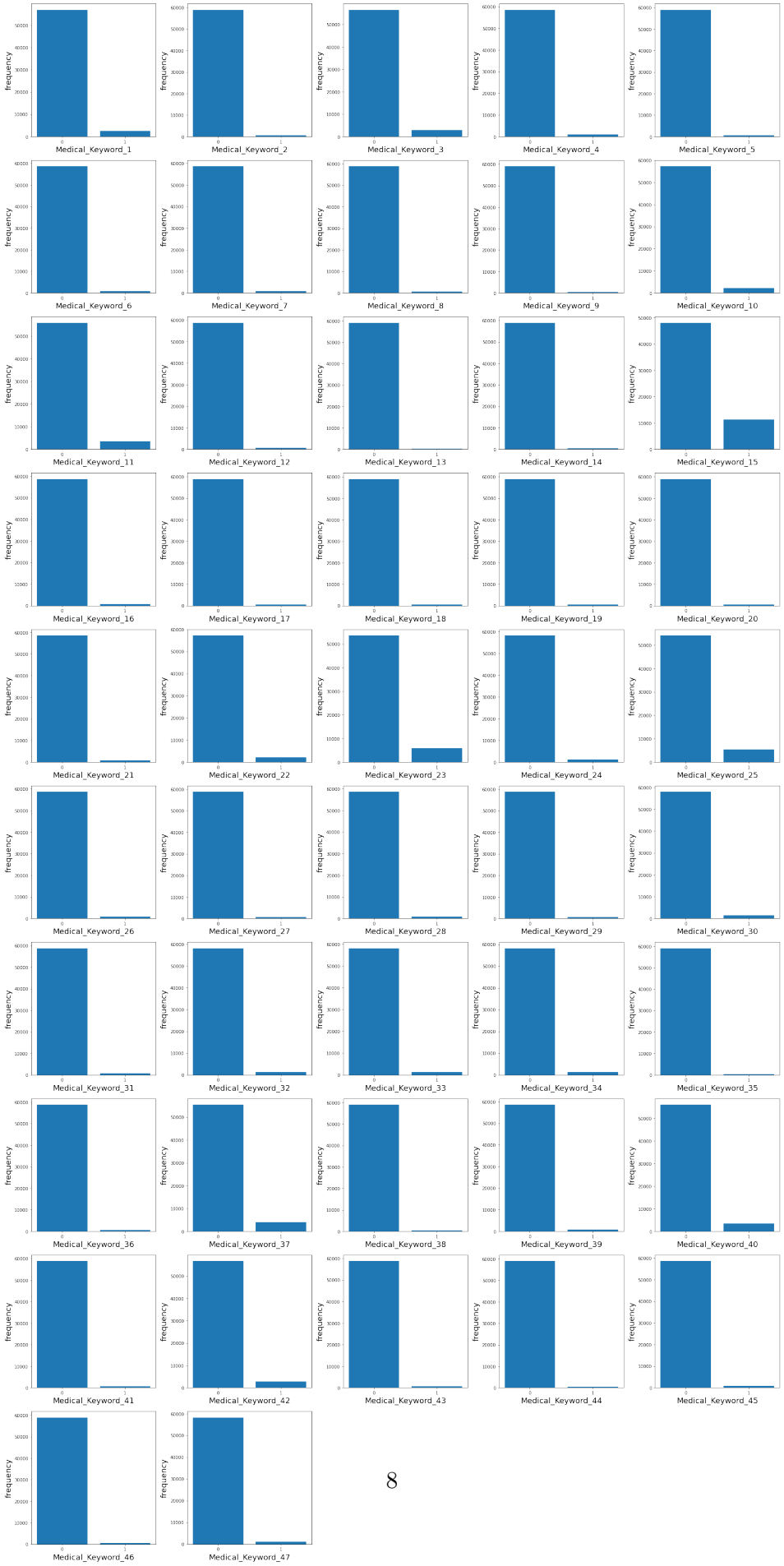
5 Discrete Variable - Plot

[20]:
```
plot_histgrams(discrete_data)
```



6 Dummy Variable - Plot

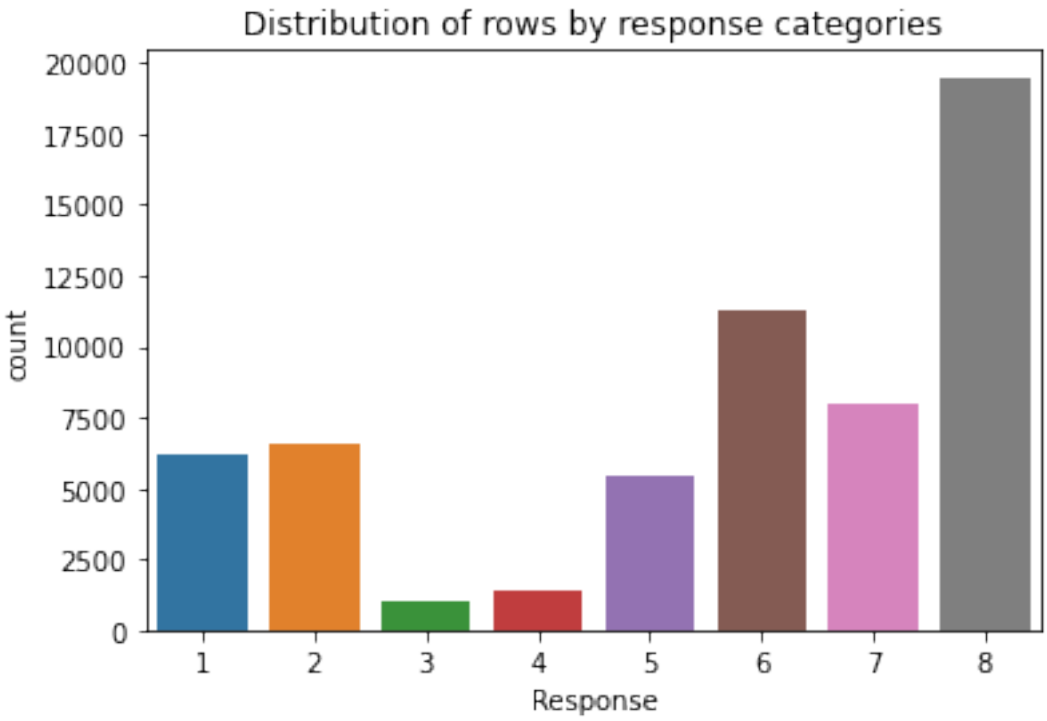
[21]:
```
plot_categoricals(dummy_data)
```


frequency
Medical_Keyword_1
Medical_Keyword_2
Medical_Keyword_3
Medical_Keyword_4
Medical_Keyword_5
Medical_Keyword_6
Medical_Keyword_7
Medical_Keyword_8
Medical_Keyword_9
Medical_Keyword_10
Medical_Keyword_11
Medical_Keyword_12
Medical_Keyword_13
Medical_Keyword_14
Medical_Keyword_15
Medical_Keyword_16
Medical_Keyword_17
Medical_Keyword_18
Medical_Keyword_19
Medical_Keyword_20
Medical_Keyword_21
Medical_Keyword_22
Medical_Keyword_23
Medical_Keyword_24
Medical_Keyword_25
Medical_Keyword_26
Medical_Keyword_27
Medical_Keyword_28
Medical_Keyword_29
Medical_Keyword_30
Medical_Keyword_31
Medical_Keyword_32
Medical_Keyword_33
Medical_Keyword_34
Medical_Keyword_35
Medical_Keyword_36
Medical_Keyword_37
Medical_Keyword_38
Medical_Keyword_39
Medical_Keyword_40
Medical_Keyword_41
Medical_Keyword_42
Medical_Keyword_43
Medical_Keyword_44
Medical_Keyword_45
Medical_Keyword_46
Medical_Keyword_47

7 Response Data Distribution

[22]:
```
sns.countplot(x=Prudential_train.Response).set_title('Distribution of rows by␣
↪response categories')
```

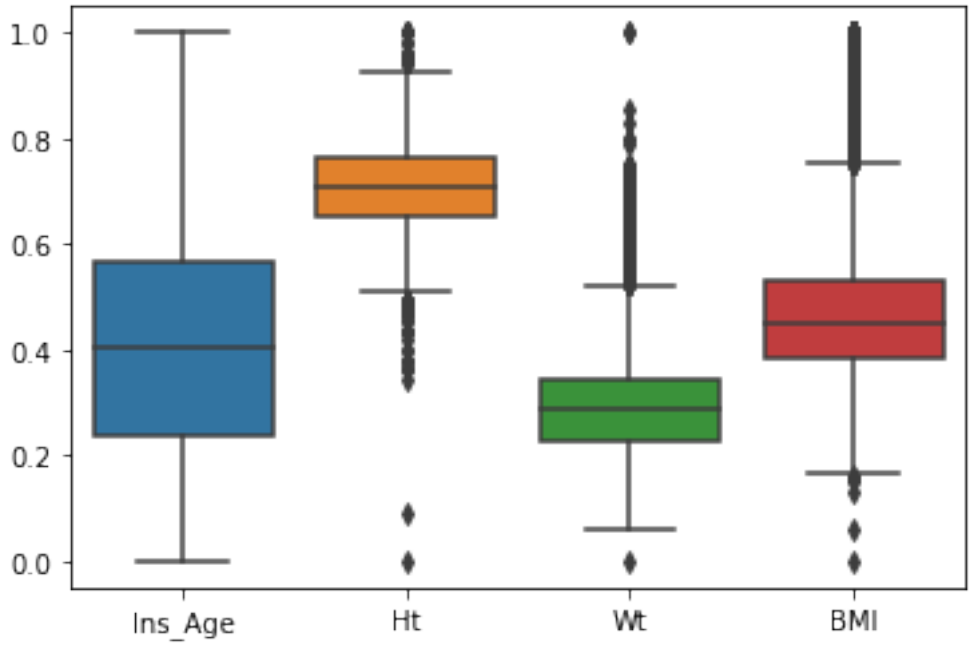
[22]:
```
Text(0.5, 1.0, 'Distribution of rows by response categories')
```



8 Outliers Plot

[23]:
```
misc_cols=["Ins_Age","Ht","Wt","BMI"]
sns.boxplot(data=Prudential_train[misc_cols])
```

[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f750b12db90>
```



9 Reassign Risk Class

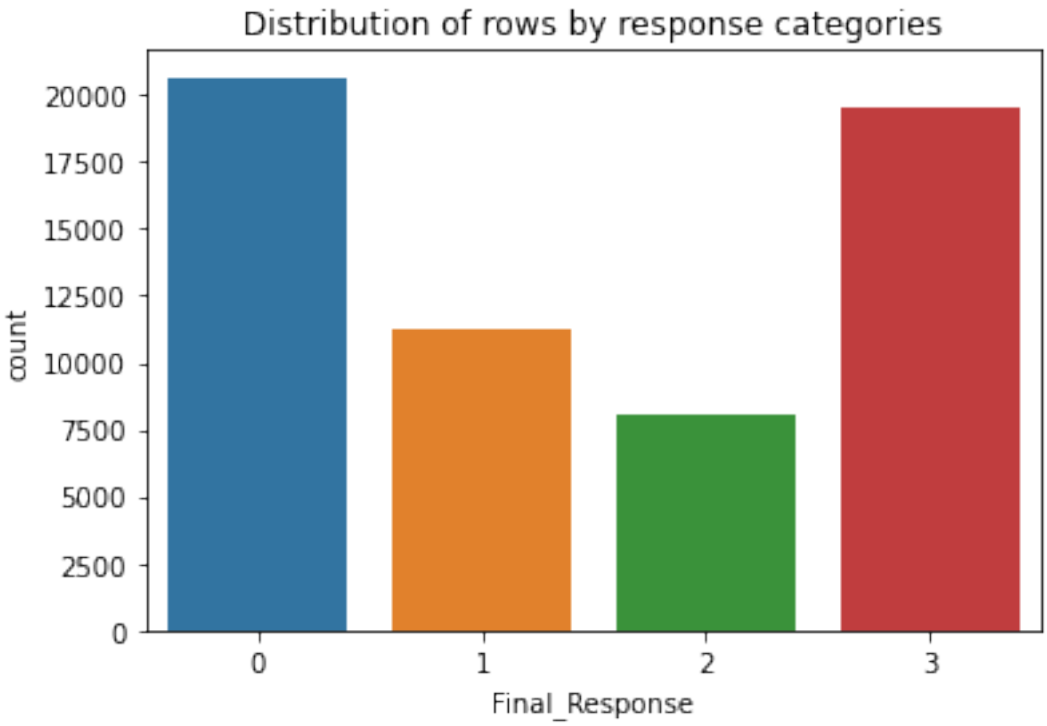
[24]:
```
prudential_train=Prudential_train.drop(axis=1,labels=["Product_Info_2"])
```

[25]:
```
prudential_train.dropna(axis=1,inplace=True)
```

[26]:
```
def new_target(row):
    if (row['Response']<=5):
        val=0
    elif (row['Response']==6):
        val=1
    elif (row['Response']==7):
        val=2
    elif (row['Response']==8):
        val=3

    else:
        val=-1
    return val
prudential_train['Final_Response']=prudential_train.apply(new_target,axis=1)
```

[27]:
```
sns.countplot(x=prudential_train.Final_Response).set_title('Distribution of␣
 ↪rows by response categories')
```

[27]: Text(0.5, 1.0, 'Distribution of rows by response categories')



10 Base Model

[28]:
```
y = prudential_train.Final_Response
X = prudential_train.drop(labels=['Response'],axis=1)
X = X.drop(labels=['Final_Response'],axis=1)
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=.
 ↪20,random_state=1)


print("Shape of X_train dataset {}".format(X_train.shape))
print("Shape of X_test dataset {}".format(X_test.shape))

print("Shape of y_train dataset {}".format(y_train.shape))
print("Shape of y_valid dataset {}".format(y_test.shape))
```

```
Shape of X_train dataset (47504, 113)
Shape of X_test dataset (11877, 113)
Shape of y_train dataset (47504,)
Shape of y_valid dataset (11877,)
```

11 Decision Tree

[29]:
```
model = DecisionTreeClassifier()
model.fit(X_train, y_train)
model_predictions = model.predict(X_test)
print("Accuracy score: {}".format(accuracy_score(y_test, model_predictions)))
print("="*80)
print(classification_report(y_test, model_predictions))
```

```
Accuracy score: 0.5117453902500632
================================================================================
              precision    recall  f1-score   support

           0       0.60      0.58      0.59      4205
           1       0.32      0.33      0.32      2206
           2       0.28      0.31      0.30      1608
           3       0.64      0.62      0.63      3858

    accuracy                           0.51     11877
   macro avg       0.46      0.46      0.46     11877
weighted avg       0.52      0.51      0.51     11877
```

12 Naive Bayes

[30]:
```
model = GaussianNB()
model.fit(X_train, y_train)
model_predictions = model.predict(X_test)
print("Accuracy score: {}".format(accuracy_score(y_test, model_predictions)))
print("="*80)
print(classification_report(y_test, model_predictions))
```

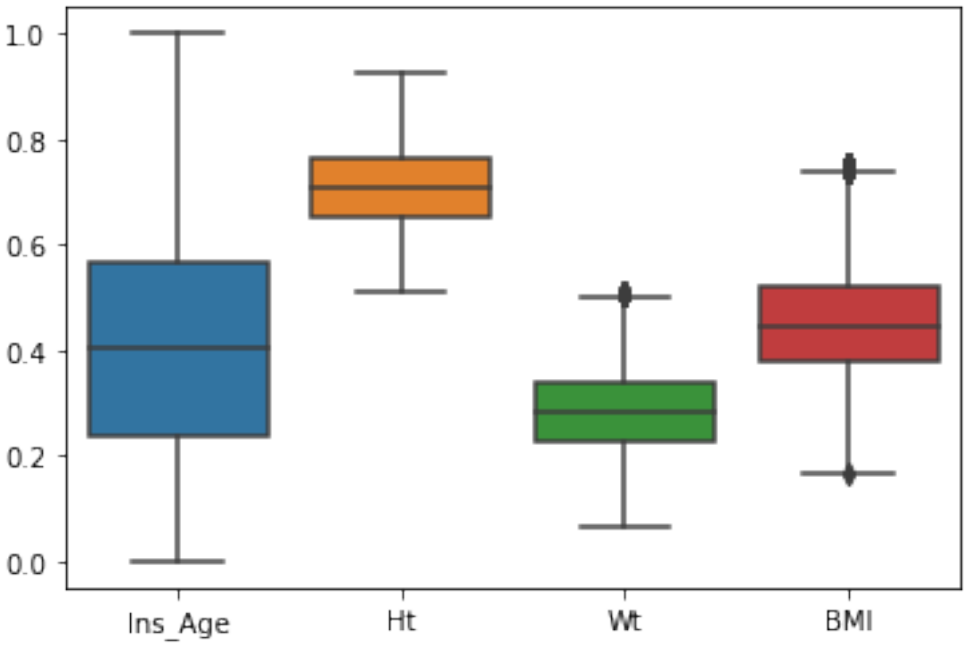
```
Accuracy score: 0.428222615138503
================================================================================
              precision    recall  f1-score   support

           0       0.71      0.21      0.33      4205
           1       0.26      0.07      0.11      2206
           2       0.29      0.22      0.25      1608
           3       0.42      0.95      0.58      3858

    accuracy                           0.43     11877
   macro avg       0.42      0.36      0.32     11877
weighted avg       0.47      0.43      0.36     11877
```

13 Treating Outliers

[31]:
```
def remove_outlier(df_in, col_name):
    q1 = df_in[col_name].quantile(0.25)
    q3 = df_in[col_name].quantile(0.75)
    iqr = q3-q1 #Interquartile range
    fence_low  = q1-1.5*iqr
    fence_high = q3+1.5*iqr
    df_out = df_in.loc[(df_in[col_name] > fence_low) & (df_in[col_name] <␣
 ↪fence_high)]
    return df_out


dev=remove_outlier(Prudential_train,'BMI')
dev=remove_outlier(dev,'Wt')
dev=remove_outlier(dev,'Ht')
```

[32]:
```
sns.boxplot(data=dev[misc_cols])
```

[32]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7503022950>
```



[33]:
```
prudential_X_train = dev
```

[34]:
```
def new_target(row):
    if (row['Response']<=5):
```

```
        val=0
    elif (row['Response']==6):
        val=1
    elif (row['Response']==7):
        val=2
    elif (row['Response']==8):
        val=3

    else:
        val=-1
    return val
prudential_X_train['Final_Response']=prudential_X_train.apply(new_target,axis=1)
```

[35]:
```
medical_keyword_cols=[col for col in prudential_X_train.columns if str(col).
 ↪startswith("Medical_Keyword")]
```

[36]:
```
medical_cols=[col for col in prudential_X_train.columns if str(col).
 ↪startswith("Medical_History")]
```

[37]:
```
prudential_X_train['Total_MedKwrds']=prudential_X_train[medical_keyword_cols].
 ↪sum(axis=1)
prudential_X_train['Total_MedHist']=prudential_X_train[medical_cols].sum(axis=1)
```

[39]:
```
prudential_X_train['Total_MedKwrds']
```

[39]:
```
0        0
1        0
2        0
3        1
4        0
        ..
59376    0
59377    0
59378    1
59379    2
59380    0
Name: Total_MedKwrds, Length: 57348, dtype: int64
```

[40]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
prudential_X_train['Product_Info_2_en'] = le.
 ↪fit_transform(prudential_X_train['Product_Info_2'])
```

[41]:
```
prudential_X_train['Product_Info_2_en']
```

[41]:
```
0        16
1         0
```

```
2        18
3        17
4        15
         ..
59376    14
59377    16
59378    18
59379    15
59380     7
Name: Product_Info_2_en, Length: 57348, dtype: int64
```

[42]:
```
prudential_X_train = prudential_X_train.drop(axis=1,labels=['Product_Info_2'])
```

[43]:
```
prudential_X_train.Final_Response.unique()
```

[43]:
```
array([3, 0, 1, 2])
```

14 Feature Selection

[44]:
```
prudential_X_train = prudential_X_train.drop(labels = ['Response'],axis=1)
prudential_X_train = prudential_X_train.drop(labels =␣
↪['Medical_History_10'],axis=1)
prudential_X_train = prudential_X_train.drop(labels = ['Id'],axis=1)
```

15 Fill Null Values

[45]:
```
prudential_X_train = prudential_X_train.fillna(prudential_X_train.mean())
```

16 Build Model

[46]:
```
y = prudential_X_train.Final_Response
X = prudential_X_train.drop(labels=['Final_Response'],axis=1)
X = X.fillna(X.mean())

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=.
↪20,random_state=1)
print("Shape of X_train dataset {}".format(X_train.shape))
print("Shape of X_test dataset {}".format(X_test.shape))

print("Shape of y_train dataset {}".format(y_train.shape))
print("Shape of y_valid dataset {}".format(y_test.shape))
```

```
Shape of X_train dataset (45878, 127)
Shape of X_test dataset (11470, 127)
Shape of y_train dataset (45878,)
Shape of y_valid dataset (11470,)
```

17 Decision Tree - with feature selection

[47]:
```
model = DecisionTreeClassifier()
model.fit(X_train, y_train)
model_predictions = model.predict(X_test)
print("Accuracy score: {}".format(accuracy_score(y_test, model_predictions)))
print("="*80)
print(classification_report(y_test, model_predictions))
```

```
Accuracy score: 0.542458587619878
================================================================================
              precision    recall  f1-score   support

           0       0.60      0.59      0.59      3731
           1       0.39      0.41      0.40      2202
           2       0.33      0.34      0.33      1642
           3       0.68      0.66      0.67      3895

    accuracy                           0.54     11470
   macro avg       0.50      0.50      0.50     11470
weighted avg       0.55      0.54      0.54     11470
```

18 Naive Bayes - with feature selection

[48]:
```
model = GaussianNB()
model.fit(X_train, y_train)
model_predictions = model.predict(X_test)
print("Accuracy score: {}".format(accuracy_score(y_test, model_predictions)))
print("="*80)
print(classification_report(y_test, model_predictions))
```

```
Accuracy score: 0.44533565823888405
================================================================================
              precision    recall  f1-score   support

           0       0.58      0.32      0.41      3731
           1       0.27      0.10      0.14      2202
           2       0.24      0.36      0.29      1642
           3       0.50      0.80      0.62      3895
```

```
    accuracy                           0.45     11470
   macro avg       0.40      0.39      0.37     11470
weighted avg       0.45      0.45      0.41     11470
```

19 Deep Neural Network

[2]:
```
!pip install tensorflow
```

```
Collecting tensorflow
  Using cached tensorflow-2.7.0-cp37-cp37m-manylinux2010_x86_64.whl (489.6 MB)
Requirement already satisfied: six>=1.12.0 in /opt/conda/lib/python3.7/site-
packages (from tensorflow) (1.14.0)
Collecting tensorboard~=2.6
  Using cached tensorboard-2.7.0-py3-none-any.whl (5.8 MB)
Requirement already satisfied: wheel<1.0,>=0.32.0 in
/opt/conda/lib/python3.7/site-packages (from tensorflow) (0.34.2)
Collecting grpcio<2.0,>=1.24.3
  Using cached
grpcio-1.41.1-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.9 MB)
Collecting tensorflow-io-gcs-filesystem>=0.21.0
  Using cached tensorflow_io_gcs_filesystem-0.22.0-cp37-cp37m-manylinux_2_12_x86
_64.manylinux2010_x86_64.whl (2.1 MB)
Collecting gast<0.5.0,>=0.2.1
  Using cached gast-0.4.0-py3-none-any.whl (9.8 kB)
Collecting absl-py>=0.4.0
  Using cached absl_py-1.0.0-py3-none-any.whl (126 kB)
Processing ./.cache/pip/wheels/3f/e3/ec/8a8336ff196023622fbcb36de0c5a5c218cbb241
11d1d4c7f2/termcolor-1.1.0-py3-none-any.whl
Collecting opt-einsum>=2.3.2
  Using cached opt_einsum-3.3.0-py3-none-any.whl (65 kB)
Requirement already satisfied: typing-extensions>=3.6.6 in
/opt/conda/lib/python3.7/site-packages (from tensorflow) (3.7.4.2)
Requirement already satisfied: protobuf>=3.9.2 in /opt/conda/lib/python3.7/site-
packages (from tensorflow) (3.11.4)
Collecting astunparse>=1.6.0
  Using cached astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Collecting flatbuffers<3.0,>=1.12
  Using cached flatbuffers-2.0-py2.py3-none-any.whl (26 kB)
Collecting wrapt>=1.11.0
  Using cached wrapt-1.13.3-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.ma
nylinux_2_12_x86_64.manylinux2010_x86_64.whl (79 kB)
Collecting tensorflow-estimator<2.8,~=2.7.0rc0
  Using cached tensorflow_estimator-2.7.0-py2.py3-none-any.whl (463 kB)
Requirement already satisfied: h5py>=2.9.0 in /opt/conda/lib/python3.7/site-
```

```
packages (from tensorflow) (2.10.0)
Collecting libclang>=9.0.1
  Using cached libclang-12.0.0-py2.py3-none-manylinux1_x86_64.whl (13.4 MB)
Collecting keras-preprocessing>=1.1.1
  Using cached Keras_Preprocessing-1.1.2-py2.py3-none-any.whl (42 kB)
Collecting keras<2.8,>=2.7.0rc0
  Using cached keras-2.7.0-py2.py3-none-any.whl (1.3 MB)
Collecting google-pasta>=0.1.1
  Using cached google_pasta-0.2.0-py3-none-any.whl (57 kB)
Requirement already satisfied: numpy>=1.14.5 in /opt/conda/lib/python3.7/site-
packages (from tensorflow) (1.18.4)
Collecting werkzeug>=0.11.15
  Using cached Werkzeug-2.0.2-py3-none-any.whl (288 kB)
Collecting markdown>=2.6.8
  Using cached Markdown-3.3.4-py3-none-any.whl (97 kB)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/opt/conda/lib/python3.7/site-packages (from tensorboard~=2.6->tensorflow)
(1.16.1)
Requirement already satisfied: requests<3,>=2.21.0 in
/opt/conda/lib/python3.7/site-packages (from tensorboard~=2.6->tensorflow)
(2.23.0)
Collecting google-auth-oauthlib<0.5,>=0.4.1
  Using cached google_auth_oauthlib-0.4.6-py2.py3-none-any.whl (18 kB)
Collecting tensorboard-plugin-wit>=1.6.0
  Using cached tensorboard_plugin_wit-1.8.0-py3-none-any.whl (781 kB)
Requirement already satisfied: setuptools>=41.0.0 in
/opt/conda/lib/python3.7/site-packages (from tensorboard~=2.6->tensorflow)
(46.1.3.post20200325)
Collecting tensorboard-data-server<0.7.0,>=0.6.0
  Using cached tensorboard_data_server-0.6.1-py3-none-manylinux2010_x86_64.whl
(4.9 MB)
Requirement already satisfied: importlib-metadata; python_version < "3.8" in
/opt/conda/lib/python3.7/site-packages (from
markdown>=2.6.8->tensorboard~=2.6->tensorflow) (1.6.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/opt/conda/lib/python3.7/site-packages (from google-
auth<3,>=1.6.3->tensorboard~=2.6->tensorflow) (0.2.8)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in
/opt/conda/lib/python3.7/site-packages (from google-
auth<3,>=1.6.3->tensorboard~=2.6->tensorflow) (4.1.0)
Requirement already satisfied: rsa<4.1,>=3.1.4 in /opt/conda/lib/python3.7/site-
packages (from google-auth<3,>=1.6.3->tensorboard~=2.6->tensorflow) (4.0)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/opt/conda/lib/python3.7/site-packages (from
requests<3,>=2.21.0->tensorboard~=2.6->tensorflow) (1.25.9)
Requirement already satisfied: idna<3,>=2.5 in /opt/conda/lib/python3.7/site-
packages (from requests<3,>=2.21.0->tensorboard~=2.6->tensorflow) (2.9)
Requirement already satisfied: certifi>=2017.4.17 in
```

```
/opt/conda/lib/python3.7/site-packages (from
requests<3,>=2.21.0->tensorboard~=2.6->tensorflow) (2020.4.5.2)
Requirement already satisfied: chardet<4,>=3.0.2 in
/opt/conda/lib/python3.7/site-packages (from
requests<3,>=2.21.0->tensorboard~=2.6->tensorflow) (3.0.4)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/opt/conda/lib/python3.7/site-packages (from google-auth-
oauthlib<0.5,>=0.4.1->tensorboard~=2.6->tensorflow) (1.3.0)
Requirement already satisfied: zipp>=0.5 in /opt/conda/lib/python3.7/site-
packages (from importlib-metadata; python_version <
"3.8"->markdown>=2.6.8->tensorboard~=2.6->tensorflow) (3.1.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in
/opt/conda/lib/python3.7/site-packages (from pyasn1-modules>=0.2.1->google-
auth<3,>=1.6.3->tensorboard~=2.6->tensorflow) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in /opt/conda/lib/python3.7/site-
packages (from requests-oauthlib>=0.7.0->google-auth-
oauthlib<0.5,>=0.4.1->tensorboard~=2.6->tensorflow) (3.0.1)
Installing collected packages: werkzeug, absl-py, markdown, grpcio, google-auth-
oauthlib, tensorboard-plugin-wit, tensorboard-data-server, tensorboard,
tensorflow-io-gcs-filesystem, gast, termcolor, opt-einsum, astunparse,
flatbuffers, wrapt, tensorflow-estimator, libclang, keras-preprocessing, keras,
google-pasta, tensorflow
Successfully installed absl-py-1.0.0 astunparse-1.6.3 flatbuffers-2.0 gast-0.4.0
google-auth-oauthlib-0.4.6 google-pasta-0.2.0 grpcio-1.41.1 keras-2.7.0 keras-
preprocessing-1.1.2 libclang-12.0.0 markdown-3.3.4 opt-einsum-3.3.0
tensorboard-2.7.0 tensorboard-data-server-0.6.1 tensorboard-plugin-wit-1.8.0
tensorflow-2.7.0 tensorflow-estimator-2.7.0 tensorflow-io-gcs-filesystem-0.22.0
termcolor-1.1.0 werkzeug-2.0.2 wrapt-1.13.3
```

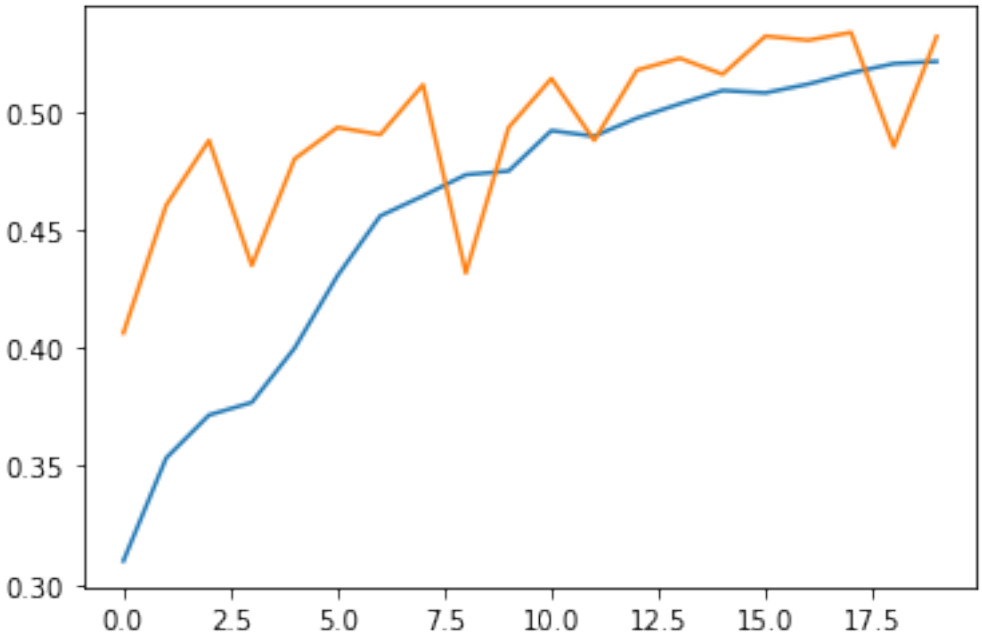
[51]:
```python
from tensorflow import keras
def get_model():
    model = keras.Sequential([
        keras.layers.Flatten(input_shape=[X.shape[-1]]),
        keras.layers.Dense(512, activation='relu'),
        keras.layers.Dense(256, activation='relu'),
        keras.layers.Dense(128, activation='relu'),
        keras.layers.Dense(64, activation='relu'),
        keras.layers.Dense(32, activation='relu'),
        keras.layers.Dropout(0.5),
        keras.layers.Dense(9, activation='softmax')
    ])
    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

[52]:
```
# DNN (fit and validation)
import tensorflow as tf

batch_size = 512
train_ds = tf.data.Dataset.from_tensor_slices((X_train.values, y_train.values)).
 ↪shuffle(len(X_train)).batch(batch_size)
val_ds = tf.data.Dataset.from_tensor_slices((X_test.values, y_test.values)).
 ↪batch(batch_size)

model = get_model()
fit = model.fit(train_ds, validation_data=val_ds, epochs=20)
```

```
Epoch 1/20
90/90 [==============================] - 2s 13ms/step - loss: 2.1382 - accuracy:
0.3099 - val_loss: 1.4310 - val_accuracy: 0.4064
Epoch 2/20
90/90 [==============================] - 1s 11ms/step - loss: 1.5067 - accuracy:
0.3535 - val_loss: 1.2871 - val_accuracy: 0.4604
Epoch 3/20
90/90 [==============================] - 1s 10ms/step - loss: 1.4167 - accuracy:
0.3716 - val_loss: 1.2783 - val_accuracy: 0.4879
Epoch 4/20
90/90 [==============================] - 1s 10ms/step - loss: 1.3693 - accuracy:
0.3770 - val_loss: 1.2494 - val_accuracy: 0.4350
Epoch 5/20
90/90 [==============================] - 1s 11ms/step - loss: 1.3209 - accuracy:
0.4000 - val_loss: 1.2194 - val_accuracy: 0.4800
Epoch 6/20
90/90 [==============================] - 1s 10ms/step - loss: 1.2899 - accuracy:
0.4305 - val_loss: 1.2015 - val_accuracy: 0.4934
Epoch 7/20
90/90 [==============================] - 1s 10ms/step - loss: 1.2658 - accuracy:
0.4559 - val_loss: 1.1972 - val_accuracy: 0.4903
Epoch 8/20
90/90 [==============================] - 1s 10ms/step - loss: 1.2517 - accuracy:
0.4644 - val_loss: 1.1863 - val_accuracy: 0.5114
Epoch 9/20
90/90 [==============================] - 1s 10ms/step - loss: 1.2370 - accuracy:
0.4733 - val_loss: 1.2382 - val_accuracy: 0.4317
Epoch 10/20
90/90 [==============================] - 1s 10ms/step - loss: 1.2350 - accuracy:
0.4749 - val_loss: 1.1683 - val_accuracy: 0.4933
Epoch 11/20
90/90 [==============================] - 1s 10ms/step - loss: 1.2046 - accuracy:
0.4920 - val_loss: 1.1334 - val_accuracy: 0.5140
Epoch 12/20
90/90 [==============================] - 1s 10ms/step - loss: 1.2119 - accuracy:
```

```
0.4896 - val_loss: 1.1680 - val_accuracy: 0.4879
Epoch 13/20
90/90 [==============================] - 1s 10ms/step - loss: 1.1981 - accuracy:
0.4973 - val_loss: 1.1245 - val_accuracy: 0.5176
Epoch 14/20
90/90 [==============================] - 1s 10ms/step - loss: 1.1842 - accuracy:
0.5034 - val_loss: 1.1073 - val_accuracy: 0.5227
Epoch 15/20
90/90 [==============================] - 1s 10ms/step - loss: 1.1688 - accuracy:
0.5090 - val_loss: 1.1394 - val_accuracy: 0.5159
Epoch 16/20
90/90 [==============================] - 1s 10ms/step - loss: 1.1695 - accuracy:
0.5080 - val_loss: 1.0976 - val_accuracy: 0.5320
Epoch 17/20
90/90 [==============================] - 1s 10ms/step - loss: 1.1567 - accuracy:
0.5117 - val_loss: 1.0920 - val_accuracy: 0.5303
Epoch 18/20
90/90 [==============================] - 1s 11ms/step - loss: 1.1517 - accuracy:
0.5165 - val_loss: 1.1308 - val_accuracy: 0.5336
Epoch 19/20
90/90 [==============================] - 1s 11ms/step - loss: 1.1430 - accuracy:
0.5204 - val_loss: 1.1563 - val_accuracy: 0.4853
Epoch 20/20
90/90 [==============================] - 1s 11ms/step - loss: 1.1405 - accuracy:
0.5213 - val_loss: 1.0746 - val_accuracy: 0.5318
```

[53]:
```python
# DNN (accuracy by epoch)
import matplotlib.pyplot as plt

plt.plot(fit.history['accuracy'])
plt.plot(fit.history['val_accuracy'])
plt.show()
```


0.50
0.45
0.40
0.35
0.30
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5