Import Tools

In [1]:
```
import os
import io
import zipfile
import numpy
import pandas
import tfdocs
import tfdocs.feed
import tfdocs.plots
import tensorflow
import numpy as np
import pandas as pd
import seaborn as sn
import matplotlib.pyplot as plt
from sklearn.model_selection import train
from sklearn.preprocessing import Imputer
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LinearSVC
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
from sklearn.model_selection import train_test_split
warnings, filterwarnings("ignore")
#variable that calls NLTK function for stemming
stemmer = nltk.stem.lancaster
#variable for stopwords removal
stopwords = nltk.corpus.stopwords.words('english')
os.environ["KMP_DUPLICATE_LIB_OK"]="True"
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python/compat/v2_compat.py:65: disable_resource_variables (from tensorflow.python.ops.variable_scope) is deprecated and will
be removed in a future version.
Instructions for updating:
non-resource variables are not supported in the long term
```

Load Data Set

In [2]:
```
#loading PIMA diabetes data set
df = pd.read_csv("diabetes_data.csv")
#creating a list of column names
cols = df.columns
#dropping rows with missing values
df.dropna(inplace=True)
```

In [3]:
```
#utility list used for holding value in pre-processing
words = []
sample = []
sample_x = []
sample_y = []
output = []
output_x = []

#open open file for training checkout
with open('diabetes.csv', 'r') :
    data = zone.read()[1:]
#iterate through first dimension of training data
for sensor in data[:500000] :
    #split sensor data by comma, is one tokenize language
    for pattern in sensor[:sensor.*] :
        #tokenize word, using nltk word_tokenizer
        words = nltk.word_tokenize(pattern)
        #end our list of words with token version
        words.append(words)
        #append corresponding word
        sample.append(words)
        #save the tag that corresponds of sample x
        sample_x.append(words[-1])
        sample_y.append(sensor[-1])
#iterate over tag has not been seen
for i in range(len(sample_x)) :
    #if we add a tag part of our labels
    if sample_x[i] not in output :
        output.append(sample_x[i])
#sort output list in line our tags
output = sorted(list(set(output))) #for w in words if w != "?"
#empty lists used
words = sorted(list(set(words)))
#sorted output list
labels = sorted(labels)
#empty training list
out_empty = [0 for _ in range(len(labels))]
#iterate through each sample and word
for x, doc in enumerate(sample_x):
    #bag/vector of utility list
    bag = []
    #stemming each word in our already stemmed examples
    wrds = [stemmer.stem(w) for w in doc]
    #iterate through each word
    #and create bag of one hot encoding
    for w in words:
        if w in wrds:
            bag.append(1)
        else:
            bag.append(0)
    #create output_row
    output_row = out_empty[:]
    output_row[output.index(sample_y[x])] = 1
    #create training data array based on encoded tokens
    training.append(bag)
    #corresponding labels for training data
    output.append(output_row)

#converting to numpy array
training = numpy.array(training)
#converting to numpy array output
output = numpy.array(output)
```

Computational Model: Pre-Processing

In [4]:
```
#utility function
#Parameters(dataframe, result_feature)
#Returns dataframe grouped by relations
#Purpose: Generate Bar Charts
    #creates dataframe, read by feature all non-empty values
    temp = df[df[pat.notnull()]]
    #groupby result feature, generate a relation between feature and result
    temp = temp[var_target].groupby([var_target,pat[var_target.name]])
    #return dataframe unstacked
    return temp
```

In [5]:
```
#utility function
#Parameters(dataframe,result_feature,median of features with result false,median of features with result true)
#Modifies the passed dataframe
def impute_pred(df,target_var,m_0,m_1):
    #iterate over columns
    for i in range(len(df)):
        #if result is false
        if df.iloc[i,target] == 0 and df.iloc[i,var] == 0:
            df.iloc[i,var] = m_0
        if df.iloc[i,target] == 1 and df.iloc[i,var] == 0:
            df.iloc[i,var] = m_1
```

In [6]:
```
#utility function
#Parameters(dataframe)
#Returns dataframe
#Purpose: Generate Bar Charts
    #suggested points by kaggle solution
    #https://www.kaggle.com/kosteg-vitya/eda-guides
    #used to separate pre-diabetic matrices with high risk diabetic matrices
    #bmi are true values
    df.loc[df['bmi'] > 0,'bmi_0'] = 1
    #description of matrix
    df.loc[df['age']<=21 & df['Outcome']==0,'A1']=0
    df.loc[df['age']>=21 & df['age']<=41 & df['Outcome']==0,'A1']=1
    df.loc[df['age']>=41 & df['Outcome']==0,'A1']=2
    #this matrix is people over 50
    df.loc[df['bmi']<18 ,'B1']=0
    df.loc[df['bmi']>=18 & df['bmi']<25 ,'B1']=1
    #this relationship between older women with high rate of pregnancies
    df.loc[df['age']>=41 & df['Pregnancies']>=6,'B2']=0
    df.loc[df['age']<=41 & df['Pregnancies']>=6,'B2']=1
    #relationship between high blood levels and cholesterol
    #high blood pressure can be a sign of many health complications
    df.loc[df['BloodPressure']>=140 & df['BloodPressure']>=90,'B3']=0
    df.loc[df['BloodPressure']<=140 & df['BloodPressure']<=90,'B3']=1
    #higher bmi results even 30 can use used to track diabetic percentage
    df.loc[df['bmi']>=30,'B4']=0
    df.loc[df['bmi']<=30 & df['bmi']>=18,'B4']=1
    #Relationship between body mass index and bodyfat
    #bmi can be a sign of signs of pre health
    df.loc[df['bmi']>=30,'B5']=0
    df.loc[df['bmi']<=30 & df['SkinThickness']>=30,'B5']=1
    df.loc[df['bmi']<=30 & df['SkinThickness']<=30,'B5']=2
    #bmi levels helps us separate high bmi patients
    df.loc[df['bmi']>=30,'B6']=0
    df.loc[df['bmi']<=30 & df['bmi']>=18,'B6']=1
    #bmi levels helps us separate high bmi patients
    #insulin level over 200 are known to be problematic by major clinic
    df.loc[df['bmi']>=30,'B7']=0
    df.loc[df['bmi']<=30 & df['bmi']>=18,'B7']=1
    #high blood pressure suggest a diastolic blood pressure to be health only if less than 80mm Hg
    df.loc[df['bmi']>=30,'B8']=0
    df.loc[df['BloodPressure']>=80,'B8']=0
    #pregnancies can result in high risk
    df.loc[df['bmi']>=30,'B9']=0
    df.loc[df['Pregnancies']>=1 & df['Pregnancies']<=4,'B9']=0
    #pregnancies can result in high risk
    df.loc[df['BP']>=80 & df['SkinThickness']>=30,'B10']=0
    df.loc[df['BP']<=80 & df['Pregnancies']>=1 ,'B10']=0
    #pregnancies and family history
    df.loc[df['BP']>=80 & df['Pregnancies']>=1 & df['DiabetesPedigreeFunction']>=1,'B11']=0
    #relationship between age and family history
    df.loc[df['BP']>=80 & df['DiabetesPedigreeFunction']>=1,'B12']=0
    #observation of bodyfat per mass index values
    df.loc[df['bmi']>=30,'B13']=0
    df.loc[df['bmi']<=30 & df['bmi']>=18,'B13']=1
    #bmi levels
    y_0 = df.loc[df['Outcome']==0,'bmi']
    y_1 = df.loc[df['Outcome']==1,'bmi']
    #dropping values
    for col in y_columns:
        df[col] = df[col].fillna(df.loc[df[target_var]==1,col])
    y = df['Outcome']
    y_columns = ['B1','B2','B3','B4','B5','B6','B7','B8','B9','B10','B11']
    #dropping unused columns
    for col in cols:
        df[col] = df[col].fillna(df.loc[df[target_var]==1,col])
    #return copy of dataset
    return y_0
```

Chatbot Model: Pre-Processing

In [7]:
```
#utility function
#Parameters(word_words list)
#Returns bag array of probability
#Purpose: Generate Bar Charts
    #bag is of words
    bag = [0*len(words) for _in range(len(words))]
    #tokenize the words
    inp_words = nltk.word_tokenize(inp)
    #stemming words
    inp_words = [stemmer.stem(word.lower()) for word in inp_words]
    #iterate through words
    for item in inp_words:
        for i, w in enumerate(words):
            if w == item:
                bag[i] = 1
    return numpy.array(bag)
```

In [8]:
```
#functional function
#Defines architecture of Keras Net
#Returns model, model_dummies structure
def build_net():
    #initialize pathes shape is 1xnxa_dim
    net = tflearn.input_data(shape=[None, len(training[0])])
    #hidden layer with 8 neurons
    net = tflearn.fully_connected(net, 8)
    #hidden layer with 8 neurons
    net = tflearn.fully_connected(net, 8)
    #activation layer used for probability of correctness
    net = tflearn.fully_connected(net, len(output[0]), activation="softmax")
    #output layer based on probability
    net = tflearn.regression(net)
    #initialize model using network
    model = tflearn.DNN(net)
    return model
```

In [9]:
```
#functional
#Trains and saves model to current directory
#returns the trained model
def train_net():
    #call to model to be built
    model = build_net()
    #iterate through based on parameters training data,labels of training data,num times training data seen,display training iterations)
    model.fit(training, output, n_epoch=1000, batch_size=8, show_metric=True)
    #save model to current directory
    model.save("model_diabetes.tflearn")
    return model
```

In [10]:
```
#utility function
#Parameters(df,pat)
#Returns dataframe of patient
#Purpose: Generate Bar Charts
def format_user_df(df,pat):
    #suggested points by kaggle solution
    #https://www.kaggle.com/kosteg-vitya/eda-guides
    #used to separate pre-diabetic matrices with high risk diabetic matrices
    #bmi are true values
    df.loc[df['bmi'] > 0,'bmi_0'] = 1
    #description of matrix
    df.loc[df['age']<=21 & df['Outcome']==0,'A1']=0
    df.loc[df['age']>=21 & df['age']<=41 & df['Outcome']==0,'A1']=1
    df.loc[df['age']>=41 & df['Outcome']==0,'A1']=2
    #this matrix is people over 50
    df.loc[df['bmi']<18 ,'B1']=0
    df.loc[df['bmi']>=18 & df['bmi']<25 ,'B1']=1
    #this relationship between older women with high rate of pregnancies
    df.loc[df['age']>=41 & df['Pregnancies']>=6,'B2']=0
    df.loc[df['age']<=41 & df['Pregnancies']>=6,'B2']=1
    #relationship between high blood levels and cholesterol
    #high blood pressure can be a sign of many health complications
    df.loc[df['BloodPressure']>=140 & df['BloodPressure']>=90,'B3']=0
    df.loc[df['BloodPressure']<=140 & df['BloodPressure']<=90,'B3']=1
    #higher bmi results even 30 can use used to track diabetic percentage
    df.loc[df['bmi']>=30,'B4']=0
    df.loc[df['bmi']<=30 & df['bmi']>=18,'B4']=1
    #Relationship between body mass index and bodyfat
    #bmi can be a sign of signs of pre health
    df.loc[df['bmi']>=30,'B5']=0
    df.loc[df['bmi']<=30 & df['SkinThickness']>=30,'B5']=1
    df.loc[df['bmi']<=30 & df['SkinThickness']<=30,'B5']=2
    #bmi levels helps us separate high bmi patients
    df.loc[df['bmi']>=30,'B6']=0
    df.loc[df['bmi']<=30 & df['bmi']>=18,'B6']=1
    #bmi levels helps us separate high bmi patients
    #insulin level over 200 are known to be problematic by major clinic
    df.loc[df['bmi']>=30,'B7']=0
    df.loc[df['bmi']<=30 & df['bmi']>=18,'B7']=1
    #high blood pressure suggest a diastolic blood pressure to be health only if less than 80mm Hg
    df.loc[df['bmi']>=30,'B8']=0
    df.loc[df['BloodPressure']>=80,'B8']=0
    #pregnancies can result in high risk
    df.loc[df['bmi']>=30,'B9']=0
    df.loc[df['Pregnancies']>=1 & df['Pregnancies']<=4,'B9']=0
    #pregnancies can result in high risk
    df.loc[df['BP']>=80 & df['SkinThickness']>=30,'B10']=0
    df.loc[df['BP']<=80 & df['Pregnancies']>=1 ,'B10']=0
    #pregnancies and family history
    df.loc[df['BP']>=80 & df['Pregnancies']>=1 & df['DiabetesPedigreeFunction']>=1,'B11']=0
    #relationship between age and family history
    df.loc[df['BP']>=80 & df['DiabetesPedigreeFunction']>=1,'B12']=0
    #observation of bodyfat per mass index values
    df.loc[df['bmi']>=30,'B13']=0
    df.loc[df['bmi']<=30 & df['bmi']>=18,'B13']=1
    #bmi levels
    y_0 = df.loc[df['Outcome']==0,'bmi']
    y_1 = df.loc[df['Outcome']==1,'bmi']
    #dropping values
    for col in y_columns:
        df[col] = df[col].fillna(df.loc[df[target_var]==1,col])
    y = df['Outcome']
    y_columns = ['B1','B2','B3','B4','B5','B6','B7','B8','B9','B10','B11']
    #dropping unused columns
    for col in cols:
        df[col] = df[col].fillna(df.loc[df[target_var]==1,col])
    #return copy of dataset
    return y_0
```

In [11]:
```
#function: live interacter
#Prompts user for input
#Prints to user input
def chat():
    #initialize
    pregnancies = []
    bp = []
    bloodpressure = []
    skinthickness = []
    insulin = []
    bmi = []
    dpf = []
    age = []
    #iterate over extracted features
    user_features = []
    #iterate over features below
    #prompt user
    for feature in ["Number of times pregnant",
                    "Plasma glucose concentration over 2 hours in an oral glucose tolerance test",
                    "Diastolic blood pressure (mm Hg)",
                    "Triceps skin fold thickness (mm)",
                    "2-Hour serum insulin (mu U/ml)",
                    "Body mass index (weight",
                    "Function which scores likelihood of diabetes based on family history",
                    "Age (years)"]:
    display information to user, may help guide
    print(pat.describe(include=['float64'])[feature_cols_matrix_mean].columns.t[-1]+" Min Mmrkrs:","How to measure")
    #get input
    print("Enter this to begin:, I can people determine if they are in risk of diabetes.")
    print("Enter this to begin:, I can help with the following information and type done to for me to calculate.")
    display(df_bin_matrix[:1])
    #initialize variable to be used in loop
    #conversation of chatbot with user
    while True:
        #enter input variable
        inp = input("You: ")
        #if user enters done ending conversation
        if inp.lower() == "done":
            #break out of loop
            break
        #iteration of connect to dmp
        #tokenize input, stem, lowercase, get bag_of_words,
        results_index = numpy.argmax(classify(inp, words))
        tag = labels[results_index]
        #
        #probabilistic extraction
        #if tag is pregnancies
        if tag == "pregnancies":
            if len(pregnancies) == 0:
                pregnancies.append(inp)
            else:
                pregnancies[0] = inp
        #if tag is "bloodpressure"
        if tag == "bloodpressure":
            if len(bp) == 0:
                bp.append(inp)
            else:
                bp[0] = inp
        #if tag is "bloodpressure"
        if tag == "bloodpressure":
            if len(bloodpressure) == 0:
                bloodpressure.append(inp)
            else:
                bloodpressure[0] = inp
        #if tag is "skinthickness"
        if tag == "skinthickness":
            if len(skinthickness) == 0:
                skinthickness.append(inp)
            else:
                skinthickness[0] = inp
        #if tag is "insulin"
        if tag == "insulin":
            if len(insulin) == 0:
                insulin.append(inp)
            else:
                insulin[0] = inp
        #if tag is "bmi"
        if tag == "bmi":
            if len(bmi) == 0:
                bmi.append(inp)
            else:
                bmi[0] = inp
        #if tag is "diabetespedigreefunction"
        if tag == "diabetespedigreefunction":
            if len(dpf) == 0:
                dpf.append(inp)
            else:
                dpf[0] = inp
        #if tag is "age"
        if tag == "age":
            if len(age) == 0:
                age.append(inp)
            else:
                age[0] = inp
        else:
        pass
        #
        for tg in data["responses"]:
            if tg['tag'] == tag:
                responses = tg['responses']
        #choose random response from intent
        print(random.choice(responses))
        #if user input is not recognized, prompt IO, please refrase your statement, otherwise ignore this message"#format(tag)
        print("\n")
        #if user mentioned feature list
        features = pregnancies+bp+bloodpressure+skinthickness+insulin+bmi+dpf+age+age
        #use not encoding for user input matrix
        for feature in features:
            temp = list(df[df['bin_'+str(i)+'_'+str(int(feature))].isnull()])
            if len(temp) > 0:
                temp = 0
            else:
                temp = temp[-1]
            user_features.append(temp)
        #duplicate of user variable
        user_df = pd.DataFrame([user_features],columns = feature_cols)
        #serialize by computational model
        user_df = format_user_df(user_df,df)
        #display to user
        if m_result == 0:
            print("You are not likely to have diabetes")
        else:
            print("You are in high risk of diabetes, please contact your medical provider")
```

Train and Validate Computational Model

In [12]:
```
# K-Fold Keras
kfold = KFold(n_splits=10, shuffle=True)
for train, test in kfold.split(df[['bmi_0','Insulin']],df.iloc[:,'Insulin']):
    x_train = df[['bmi_0','Insulin']]

x_train, x_validate, y_train, y_validate = train_test_split(x, y,test_size = 2,random_state=0)

bagging = BaggingClassifier(
    base_estimator=DecisionTreeClassifier(),
    n_estimators=100,max_samples=100,
    bootstrap=True,bootstrap_features=False,
    oob_score= True)

bagging.fit(x_train,y_train)

print(classification_report(bagging.predict(x_validate),y_validate))
```

```
              precision    recall  f1-score   support

           0       0.94      0.98      0.96      [illegible]
           1       0.94      0.88      0.90      [illegible]

    accuracy                           0.94      [illegible]
   macro avg       0.94      0.93      0.93      [illegible]
weighted avg       0.94      0.94      0.94      [illegible]
```

Test Chatbot Model

In [13]:
```
model = train_net()
```

```
Training Step: 8000  | total loss: 0.00013 | time: 0.010s
| Adam | epoch: 1000 | loss: 0.00013 - acc: 0.9997 -- iter: 48/53
Training Step: 7999  | total loss: 0.00014 | time: 0.010s
| Adam | epoch: 1000 | loss: 0.00014 - acc: 0.9997 -- iter: 53/53

INFO:tensorflow:/home/ec2-user/diabetes.Checkpoint/checkpoint.tfpack is not in all_model_checkpoint_paths. Manually adding it.
```

In [14]:
```
chat()
```

```
Hello this is bugbot, I can people determine if they are in risk of diabetes.
Enter this to begin:, I can help with the following information and type done to for me to calculate:
                 Bmi_0Insulin
count  [illegible]  [illegible]
mean   [illegible]  [illegible]
std    [illegible]  [illegible]
min    [illegible]  [illegible]
25%    [illegible]  [illegible]
50%    [illegible]  [illegible]
75%    [illegible]  [illegible]
max    [illegible]  [illegible]
Number of times pregnancies Min Mmrkrs: How to measure
Plasma glucose concentration over 2 hours in a... Min Mmrkrs: How to measure
Diastolic blood pressure (mm Hg) Min Mmrkrs: How to measure
Triceps skin fold thickness (mm) Min Mmrkrs: How to measure
2-Hour serum insulin (mu U/ml) Min Mmrkrs: How to measure
Body mass index (weight Min Mmrkrs: How to measure
Function which scores likelihood of diabetes based on family history Min Mmrkrs: How to measure
Age (years) Min Mmrkrs: How to measure
Enter this to begin:, I can people determine if they are in risk of diabetes.
Enter this to begin:, I can help with the following information and type done to for me to calculate.
   Pregnancies  ...  Age
0            1  ...   [illegible]

[1 rows x 8 columns]
You: done
Thank you for using our chatbot.
Enter, thank you for informing me with your pregnancies
If you're refreshing to your Pregnancies, please refrase your statement, otherwise ignore this message

You: Sugar levels are 140
Thank you for informing me with of your glucose levels
If you're refreshing to your Glucose, please refrase your statement, otherwise ignore this message

You: Insulin is 175
Thank you for informing us with your insulin levels
If you're refreshing to your Insulin, please refrase your statement, otherwise ignore this message

You: I am 30 years old
Thank you for providing us with your age
If you're refreshing to your Age, please refrase your statement, otherwise ignore this message

You: My BMI is about 40
Thank you for providing us with your BMI
If you're refreshing to your BMI, please refrase your statement, otherwise ignore this message

You: I don't know my pedigree function
Thank for providing us with your diabetes pedigree value
If you're refreshing to your DiabetesPedigreeFunction, please refrase your statement, otherwise ignore this message

You: Skin Thickness is 35
Thank you for providing us with skin thickness
If you're refreshing to your SkinThickness, please refrase your statement, otherwise ignore this message

You: Blood Pressure is 72
Thank you for your blood pressure readings
If you're refreshing to your BloodPressure, please refrase your statement, otherwise ignore this message

You: done
You are in high risk of diabetes, please contact your medical provider
```

In [15]:
```
```