Assignment 8

Using any dataset that has a categorical feature that needs to be predicted, use several algorithms, preprocessing techniques, feature extraction techniques to fit the data to the model and show the accuracy, confusion matrix, and the classification report. G
https://www.kaggle.com/ntnu-testimon/paysim1
https://www.kaggle.com/joniarroba/noshowappointments
https://archive.ics.uci.edu/ml/datasets.html?format=&task=cla&att=&area=&numAtt=&numIns=&type=&sort=nameUp&view=table
https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients
https://archive.ics.uci.edu/ml/datasets/Adult

I selected: https://www.kaggle.com/joniarroba/noshowappointments

In [1]:
```
#Packages
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set(font_scale=1.5)
from sklearn import svm, datasets, preprocessing
from sklearn.preprocessing import scale, LabelEncoder, OneHotEncoder, s
cale
from sklearn.metrics import accuracy_score, confusion_matrix, classific
ation_report
import pandas as pd
from pylab import rcParams
rcParams['figure.figsize'] = 20,10

#Models
from sklearn.svm import SVC as Model_1
from sklearn.svm import LinearSVC as Model_2
```

```
from sklearn.neighbors import KNeighborsClassifier as Model_3
from sklearn.linear_model import LogisticRegression as Model_4
```

In [2]:
```
#open dataset, downloaded from https://www.kaggle.com/joniarroba/noshow appointments
df = pd.read_csv(r'C:\Users\carlb\Desktop\KaggleV2-May-2016.csv')
df.dropna(inplace=True)
```

Preprocessing

In [3]:
```
#Get Dummies:

#Gender
df = pd.concat([df,pd.get_dummies(df['Gender'], prefix='Gender',dummy_na=True)],axis=1)

#No-show
df = pd.concat([df,pd.get_dummies(df['No-show'], prefix='No-show',dummy_na=True)],axis=1)
#SMS_received
df = pd.concat([df,pd.get_dummies(df['SMS_received'], prefix='SMS_received',dummy_na=True)],axis=1)
```

In [4]:
```
#Binary object-columns to int64

#Gender
df['Gender'] = df['Gender'].replace('F','0')
df['Gender'] = df['Gender'].replace('M','1.0')
df["Gender"] = pd.to_numeric(df["Gender"])

#SMS_received
df['SMS_received'] = df['SMS_received'].replace('Yes','1.0')
df['SMS_received'] = df['SMS_received'].replace('No','0')
df["SMS_received"] = pd.to_numeric(df["SMS_received"])

#No-Show
```

```
df['No-show'] = df['No-show'].replace('Yes','1.0')
df['No-show'] = df['No-show'].replace('No','0')
df["No-show"] = pd.to_numeric(df["No-show"])
```

In [5]:
```
#Date-objects to datetime format

#ScheduledDay
df['parsed_ScheduledDay'] = pd.to_datetime(df.ScheduledDay)

#AppointmentDay
df['parsed_AppointmentDay'] = pd.to_datetime(df.AppointmentDay)
```

Feature Selection

In [6]:
```
#Assignment of Month Variable
df['month'] = df.parsed_ScheduledDay.dt.month
```

In [7]:
```
#Assignment of Day of week Variable
df['dayofweek'] = df.parsed_ScheduledDay.dt.dayofweek
```

In [8]:
```
#Time between scheduled day and appointment day:

#(1)Calculate difference
#df['time_between'] = df['parsed_AppointmentDay'] - df['parsed_Schedule
dDay']

#(2)convert to int64
#df['time_between'] = df['time_between'] / np.timedelta64(1, 's') / 864
00

#(3)convert same-day-scheduled (neg values to positive) & to float64

#I did this block in excel. Couldn't wrap my head around the writing th
e function to convert neg values to postivive in python.
```

In [9]:
```
df.dtypes
```

Out[9]:
```
PatientId                       float64
AppointmentID                     int64
Gender                          float64
ScheduledDay                     object
AppointmentDay                   object
Age                               int64
Neighbourhood                    object
Scholarship                       int64
Hipertension                      int64
Diabetes                          int64
Alcoholism                        int64
Handcap                           int64
SMS_received                      int64
No-show                         float64
time_between                      int64
Gender_F                          uint8
Gender_M                          uint8
Gender_nan                        uint8
No-show_No                        uint8
No-show_Yes                       uint8
No-show_nan                       uint8
SMS_received_0.0                  uint8
SMS_received_1.0                  uint8
SMS_received_nan                  uint8
parsed_ScheduledDay      datetime64[ns]
parsed_AppointmentDay    datetime64[ns]
month                             int64
dayofweek                         int64
dtype: object
```

Fun Part:

SVC (would run for 20 mins then stall, according to Task Manager)

In [ ]:
```
#Model 1 (SVC)
```

```
feature_cols = ['time_between','Age','SMS_received']
X = df[feature_cols]
y = df['No-show']

model = Model_1()
model.fit(X,y)

df['pred_1'] = model.predict(X)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(df['No-show'], df.pred_1.round
()))
print(" ")
print("Confustion Matrix:")
print(confusion_matrix(df['No-show'], df.pred_1.round()))
print(" ")
print("Classification Report:")
print(classification_report(df['No-show'], df.pred_1.round()))
print("----------------------------------------------------------------
-------------")
```

Linear SVC

In [10]:
```
#Model 2 (Linear SVC)
feature_cols = ['time_between','Age','SMS_received']
X = df[feature_cols]
y = df['No-show']

model = Model_2()
model.fit(X,y)

df['pred_2'] = model.predict(X)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(df['No-show'], df.pred_2.round
()))
print(" ")
print("Confustion Matrix:")
```

```
print(confusion_matrix(df['No-show'], df.pred_2.round()))
print(" ")
print("Classification Report:")
print(classification_report(df['No-show'], df.pred_2.round()))
print("------------------------------------------------------------
-------------")
```

```
Accuracy Score: 0.7977598233915695

Confustion Matrix:
[[88141    67]
 [22286    33]]

Classification Report:
             precision    recall  f1-score   support

        0.0       0.80      1.00      0.89     88208
        1.0       0.33      0.00      0.00     22319

avg / total       0.70      0.80      0.71    110527

------------------------------------------------------------------
------
```

Nearest Neighbors

In [11]:
```
#Model 3 (Nearest Neighbors)
feature_cols = ['time_between','Age','SMS_received']
X = (df[feature_cols])
y = df['No-show']

model = Model_3(2)
model.fit(X,y)

df['pred_3'] = model.predict(X)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(df['No-show'], df.pred_3.round
```

```
()))
print(" ")
print("Confustion Matrix:")
print(confusion_matrix(df['No-show'], df.pred_3.round()))
print(" ")
print("Classification Report:")
print(classification_report(df['No-show'], df.pred_3.round()))
print("-----------------------------------------------------------------
-------------")
```

```
Accuracy Score: 0.7979860124675419

Confustion Matrix:
[[84696  3512]
 [18816  3503]]

Classification Report:
             precision    recall  f1-score   support

        0.0       0.82      0.96      0.88     88208
        1.0       0.50      0.16      0.24     22319

avg / total       0.75      0.80      0.75    110527

------------------------------------------------------------------------
------
```

In [12]:
```
#Model 4 (Logistic Regression)
feature_cols = ['time_between','Age','SMS_received']
X = df[feature_cols]
y = df['No-show']

model = Model_4()
model.fit(X,y)

df['pred_4'] = model.predict(X)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(df['No-show'], df.pred_4.round
```

```
()))
print(" ")
print("Confustion Matrix:")
print(confusion_matrix(df['No-show'], df.pred_4.round()))
print(" ")
print("Classification Report:")
print(classification_report(df['No-show'], df.pred_4.round()))
```

```
Accuracy Score: 0.795027459353823

Confustion Matrix:
[[87566   642]
 [22013   306]]

Classification Report:
             precision    recall  f1-score   support

        0.0       0.80      0.99      0.89     88208
        1.0       0.32      0.01      0.03     22319

avg / total       0.70      0.80      0.71    110527
```

In [13]:
```
df.to_csv(r'C:\Users\carlb\Desktop\model_out.csv')
```