Group 03: CopyPaste

Datasets used: Dataset_Hospital_Vists.csv, test.csv

Participants:

- Gina-Maria Tanja Färber 77211826704
- Muhammad Raza 77211990025
- Ligia Vergara 77212002271

Contents Draft

1. Problem and Goal Definition

Problem: Patients in a hospital miss their scheduled appointments.

Goal: Develop a machine learning model that predicts if a patient will miss a future appointment.

2. Data Understanding

2.1 Dataset Description

- The dataset is at appointment level granularity and contains detail of each appointment and patient.
- It has 14 columns of which 1 will be our target variable: No-show.
- We mostly have information about an appointment's date and place and the patients' health details. A column also shows if a patient received an SMS before the appointment.

2.2 Quick Analysis from Kaggle

- There is missing data in the columns Age, Community, Social Welfare, and some diseases.

- Female to Male ratio is 65:35.
- We have no NULLs in the target variable.
- For Handcap, we have multiple values even though it seems to be a binary variable.

3. Data Quality Check

We check our dataset against the following dimensions:

- Uniqueness
- Missing data
- Data type consistency check
- Distribution of Categorical Variables
- Dates inconsistency

In [ ]:
```
# Setting up environment with packages

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

In [ ]:
```
# Importing the dataset

df = pd.read_csv("/Users/muhammadraza/Documents/GitHub/BIPM/Data Science/
df.head()

# Increase seaborn default resolution

sns.set(rc={"figure.dpi":150, 'savefig.dpi':150})
sns.set_context('notebook')
sns.set_style("ticks")
sns.set(rc={'figure.figsize':(5,6)})

# Give variables to color numbers

green = '#008000'
red = '#ff0000'
```

In [ ]:
```
## Uniqueness

# Is each row unique?

print("Duplicate rows: " + str(df.duplicated().sum()))

# Is each appointmentID unique?

print("Duplicate appointments: " + str(df['AppointmentID'].duplicated().s
```

```
Duplicate rows: 0
Duplicate appointments: 0
```

We can conclude that the dataset only containts unique IDs and no duplicates.

In [ ]:
```
## Missing Data

# Which columns have missing data?

missing_data = df.isnull().sum()
total_entries = len(df)
percentage_missing = round((missing_data / total_entries) * 100, 2)

missing_info = pd.DataFrame({
    'Missing Count': missing_data,
    'Percentage Missing': percentage_missing
})

print(missing_info)
```

```
                 Missing Count  Percentage Missing
PatientId                    0                0.00
AppointmentID                0                0.00
Sex                          0                0.00
ScheduledDate                0                0.00
AppointmentDate              0                0.00
Age                       8807                9.96
Community                10713               12.12
SocialWelfare            12519               14.16
Hipertension              8021                9.07
Diabetes                     0                0.00
Alcoholism               14889               16.84
Handcap                      0                0.00
SMS_received                 0                0.00
No-show                      0                0.00
```

As also seen from Kaggle, Age, Community, SocialWelfare, Hipertension, and Alcoholism have significant null values.

In [ ]:
```
## Data Type Consistency

df.dtypes
```

Out[ ]:
```
PatientId          float64
AppointmentID        int64
Sex                 object
ScheduledDate       object
AppointmentDate     object
Age                float64
Community           object
SocialWelfare       object
Hipertension        object
Diabetes            object
Alcoholism          object
Handcap             object
SMS_received        object
No-show             object
dtype: object
```

1. ScheduledDate and AppointmentDate must be timestamps and not objects.

In [ ]:
```
# Distribution of Categorical Variables

```

```
occ = df.groupby('Handcap').size().reset_index()
print(occ)
```

```
  Handcap      0
0       2    139
1       3     11
2       4      3
3      no  86626
4     yes   1642
```

We assume that this column was meant to be a binary column and the numerical values are bad data. They will be converted to categorical (yes) during preprocessing. The assumption here is that someone entered the number of handicaps to a person rather than a yes or a no.

In [ ]:
```
# Dates inconsistency

from datetime import datetime

df['AppointmentDate'] = df['AppointmentDate'].apply(lambda x: datetime.st
df['ScheduledDate'] = df['ScheduledDate'].apply(lambda x: datetime.strpti

counts = df['ScheduledDate'] > df['AppointmentDate']
occurrence_counts = counts.value_counts()

print(occurrence_counts)
```

```
False    88417
True         4
Name: count, dtype: int64
```

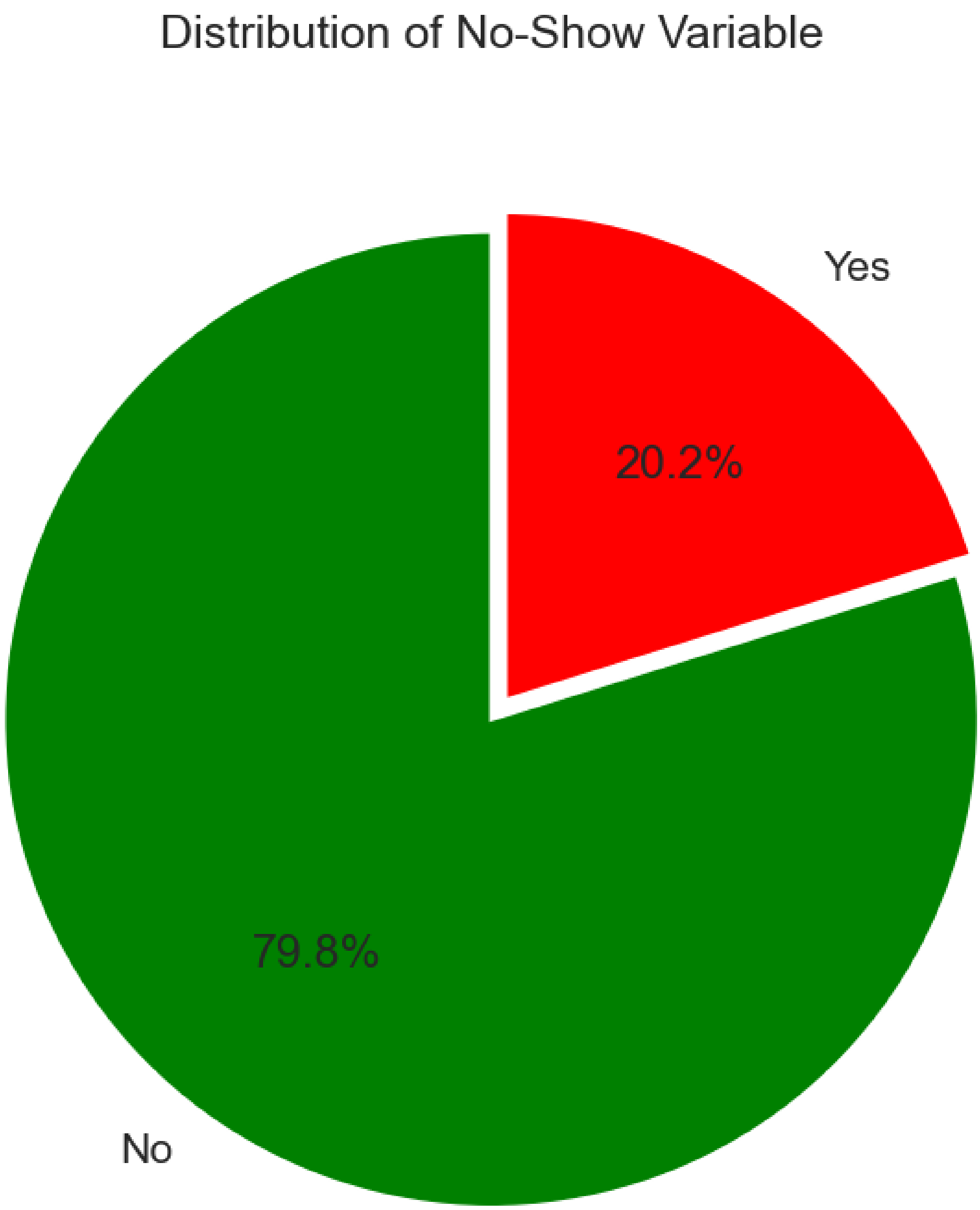
We see that we have 4 incoherent combination of schedule and appointment dates - they will be taken out in data cleaning stage.

4. Exploratory Data Analysis

- Distribution of the target variable.
- Distribution of age. Which age groups account for the most missing appointments?
- Do patients of a certain community miss their appointments more than others?
- Do males or females miss more appointments?

In [ ]:
```
## Distribution of the target variable

value_counts = df['No-show'].value_counts()

labels = value_counts.index
sizes = value_counts.values

colors = [green,red]  # Customize colors
explode = (0.05, 0)  # Explode the 1st slice

plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangl

plt.title('Distribution of No-Show Variable')
plt.axis('equal')  # Equal aspect ratio ensures the pie chart is circular
```

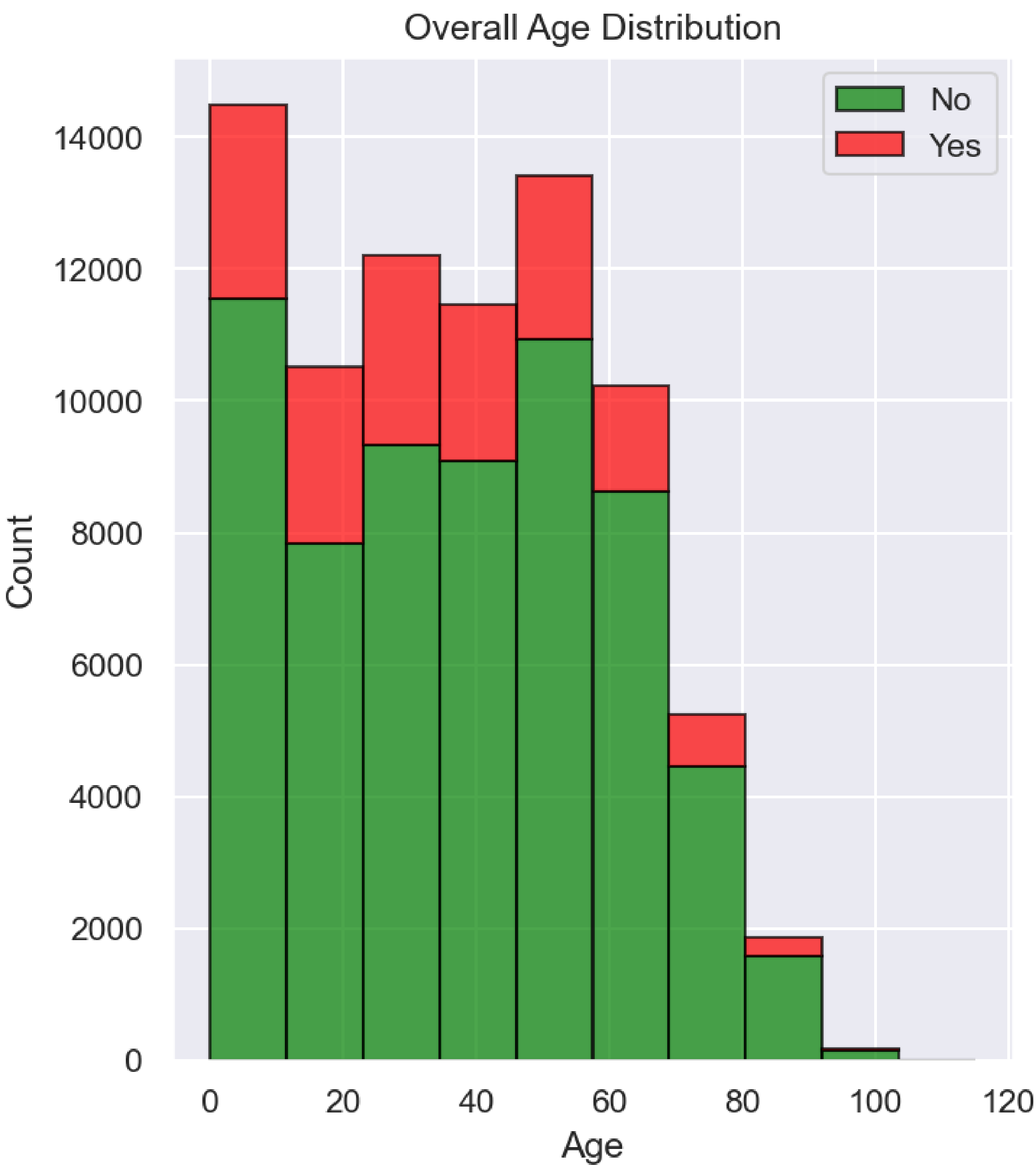
```
plt.show()
```



1 in 5 appointments are missed on average.

In [ ]:
```
# Which age group misses more appointments?

# Plotting histogram with split bars

plt.hist([df[df['No-show'] == 'No']['Age'], df[df['No-show'] == 'Yes']['A
         bins=10, color=['green', 'red'], alpha=0.7, edgecolor='black', l

plt.title('Overall Age Distribution')
plt.xlabel('Age')
plt.ylabel('Count')
plt.legend()
```

```
plt.show()
```

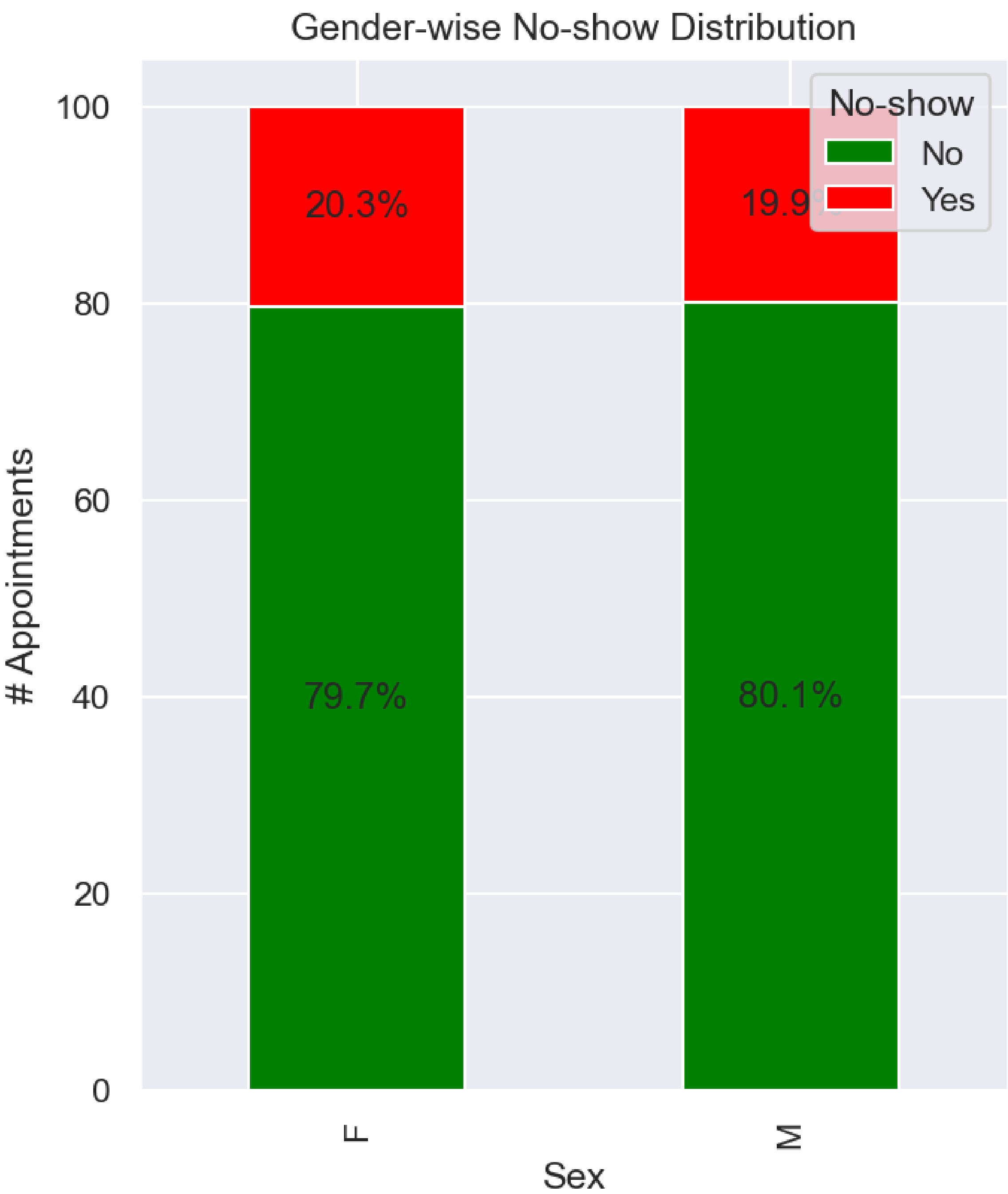


There is an even distribution of the ratio of appointments missed in the various age groups. This tends to change after age 70 where appointments are missed a lot less.

This could be explained by the fact that older people cannot afford to miss appointments due to more serious health issues and due to the fact that they might have more time on their hand.

In [ ]:
```
## Which gender misses more appointments?

# Grouping by 'gender' and 'no_show' and count occurrences
grouped_data = df.groupby(['Sex', 'No-show']).size().unstack()

# Calculating percentages

percentages = grouped_data.div(grouped_data.sum(axis=1), axis=0) * 100
```

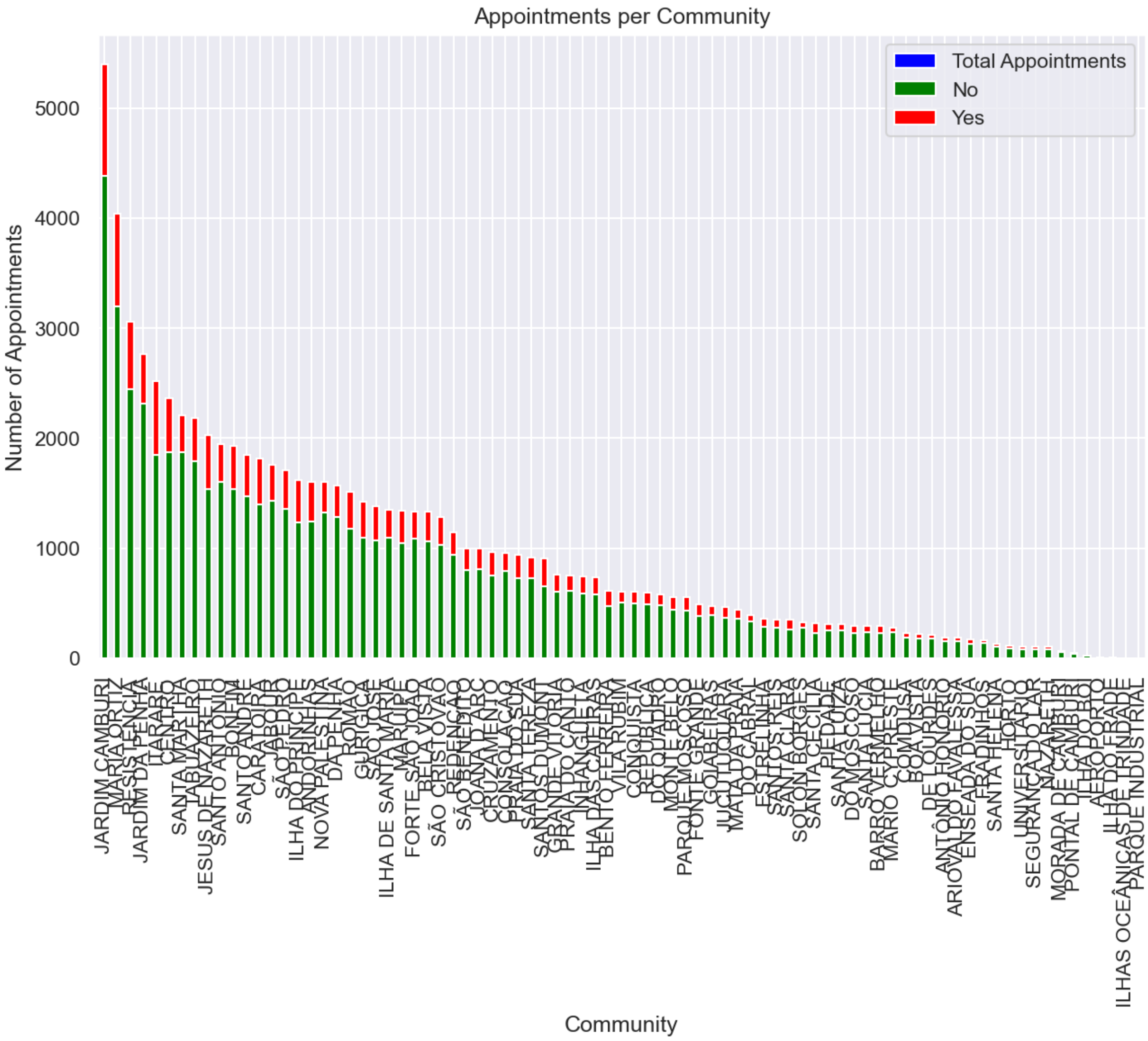
```
# Plotting a grouped bar chart

ax = percentages.plot(kind='bar', stacked=True, color=[green, red])

# Annotating bars with percentages

for p in ax.patches:
    width, height = p.get_width(), p.get_height()
    x, y = p.get_xy()
    ax.annotate(f'{height:.1f}%', (x + width/2, y + height/2), ha='center

plt.title('Gender-wise No-show Distribution')
plt.xlabel('Sex')
plt.ylabel('# Appointments')
plt.legend(title='No-show', loc='upper right')

plt.show()
```



There seems to be 1 in 5 appointments missed for both genders.

In [ ]:
```
# Do patients of a certain community miss their appointments more than ot

# Counting the total number of appointments per community
total_appointments_per_community = df['Community'].value_counts()

# Sorting the DataFrame based on the total number of appointments

sorted_df = df[df['Community'].isin(total_appointments_per_community.inde
sorted_df['Community'] = pd.Categorical(sorted_df['Community'], categorie
sorted_df = sorted_df.sort_values(by=['Community'])

# Counting the number of appointments per community split by show_up stat

appointments_per_community_show_up = sorted_df.groupby(['Community', 'No-

# Plotting the bar chart

fig, ax = plt.subplots(figsize=(10, 6))

# Bar chart for total appointments per community

total_appointments_per_community.loc[sorted_df['Community'].unique()].plo

# Bar chart for appointments per community split by show_up status

appointments_per_community_show_up.plot(kind='bar', stacked=True, ax=ax,

# Adding labels and legend

ax.set_title('Appointments per Community')
ax.set_xlabel('Community')
ax.set_ylabel('Number of Appointments')
ax.legend()

plt.show()
```

```
/var/folders/1z/kkxxkq_90qldq3ngyx1pj0fr0000gn/T/ipykernel_42276/37031659
0.py:14: FutureWarning: The default of observed=False is deprecated and wi
ll be changed to True in a future version of pandas. Pass observed=False t
o retain current behavior or observed=True to adopt the future default and
silence this warning.
  appointments_per_community_show_up = sorted_df.groupby(['Community', 'No
-show']).size().unstack(fill_value=0)
```



No community shows significant difference from the average ratio of missing appointments.

5. Data Cleaning and Basic Preprocessing

- Dropping erroneous data
- Substituting illogical entries in Handcap column
- Extrapolating data for each patient where exists
- Time between ScheduledDate and AppointmentDate
- Standardization of Continuous variable

In [ ]:
```
# Keeping original df intact

df_t = df[:]
```

In [ ]:
```
# Dropping erroneous data: Inconsistent dates from transformed dataframe.
# a negative difference would mean that we could run into processing erro

df_t = df_t.drop(df[df['ScheduledDate'] > df['AppointmentDate']].index)
```

In [ ]:
```
# Assuming that the number is the number of handicaps

df_t.loc[df_t['Handcap'].isin(['2','3','4']), 'Handcap'] = 'yes'
```

```
occ = df_t.groupby('Handcap').size().reset_index()
print(occ)
```

```
  Handcap      0
0      no  86623
1     yes   1794
```

In [ ]:
```
# Deriving New Feature: Time between ScheduledDate and AppointmentDate. W

from datetime import timedelta

df_t['time_bw_schedule_appointment'] = df_t['AppointmentDate'] - df_t['Sc

## Convert to float (days)

df_t['time_bw_schedule_appointment'] = df_t['time_bw_schedule_appointment
df_t['time_bw_schedule_appointment'] = df_t['time_bw_schedule_appointment
```

In [ ]:
```
# Extrapolating missing data: if a patient has non-null attributes for on
# Since our data is within the scope of one year, we do not need to worry

missing_columns = ['Age', 'Community', 'SocialWelfare', 'Hipertension', '

for column in missing_columns:
    df_t[column] = df_t.groupby('PatientId')[column].transform(lambda x:
```

```
/var/folders/1z/kkxxkq_90qldq3ngyx1pj0fr0000gn/T/ipykernel_42276/134371395
7.py:7: FutureWarning: Series.fillna with 'method' is deprecated and will
raise in a future version. Use obj.ffill() or obj.bfill() instead.
  df_t[column] = df_t.groupby('PatientId')[column].transform(lambda x: x.f
illna(method='ffill').fillna(method='bfill'))
```

In [ ]:
```
# Capitalising yes/no so they can be converted to a binary column.

df_t = df_t.applymap(lambda x: x.capitalize() if isinstance(x, str) else
missing_data = df_t.isnull().sum()
total_entries = len(df_t)
percentage_missing = round((missing_data / total_entries) * 100, 2)
```

```
/var/folders/1z/kkxxkq_90qldq3ngyx1pj0fr0000gn/T/ipykernel_42276/89469841
0.py:3: FutureWarning: DataFrame.applymap has been deprecated. Use DataFra
me.map instead.
  df_t = df_t.applymap(lambda x: x.capitalize() if isinstance(x, str) else
x)
```

In [ ]:
```
# Checking for remaining missing data after extrapolation: there is still
# doing it after the train-test split to avoid target leakage.

missing_info = pd.DataFrame({
    'Missing Count': missing_data,
    'Percentage Missing': percentage_missing
})

print(missing_info)
```

```
                             Missing Count  Percentage Missing
PatientId                                0                0.00
AppointmentID                            0                0.00
Sex                                      0                0.00
ScheduledDate                            0                0.00
AppointmentDate                          0                0.00
Age                                   3778                4.27
Community                             4631                5.24
SocialWelfare                         5450                6.16
Hipertension                          3406                3.85
Diabetes                                 0                0.00
Alcoholism                            6611                7.48
Handcap                                  0                0.00
SMS_received                             0                0.00
No-show                                  0                0.00
time_bw_schedule_appointment             0                0.00
```

6. Train-Test Split

In [ ]:
```
from sklearn.model_selection import train_test_split

# Avoiding Target Leakage by ensuring each patient is only in one dataset

unique_patient_ids = df_t['PatientId'].unique()

# Splitting into test and train set.

patients_train, patients_test = train_test_split(unique_patient_ids, test

# Removing the unnecessary columns as they are not needed anymore: both a
# We do not need Patient and Appointment IDs (trivial).
# No-show is the target variable which has to be removed from training.
# Scheduled and Appointment Dates are not needed as we have the differenc

columns_to_drop = ["PatientId", "No-show", "AppointmentID", "ScheduledDat

# Splitting into train and test

X_train = df_t[df_t['PatientId'].isin(patients_train)].drop(columns_to_dr
y_train = df_t[df_t['PatientId'].isin(patients_train)]["No-show"]

X_test = df_t[df_t['PatientId'].isin(patients_test)].drop(columns_to_drop
y_test = df_t[df_t['PatientId'].isin(patients_test)]["No-show"]
```

In [ ]:
```
## Checking Target variable ratio in both groups to handle potential clas

train_counts = y_train.value_counts(normalize=True) * 100
train_counts
```

Out[ ]:
```
No-show
No     79.827094
Yes    20.172906
Name: proportion, dtype: float64
```

In [ ]:
```
## Checking Target variable in both groups

test_counts = y_test.value_counts(normalize=True) * 100
test_counts
```

Out[ ]:
```
No-show
No     79.77209
Yes    20.22791
Name: proportion, dtype: float64
```

In [ ]:
```
# Getting an overview of how X_train looks like

X_train.head()
```

Out[ ]:

	Sex	Age	Community	SocialWelfare	Hipertension	Diabetes	Alcoholism	Hanc
0	F	24.0	Resistência	No	No	No	No	
2	F	19.0	Jardim da penha	No	No	No	No	
3	F	55.0	Jesus de nazareth	No	Yes	No	No	
5	F	51.0	Maruípe	No	Yes	No	No	
8	F	NaN	Santos dumont	NaN	No	No	No	

7. Advanced Data Preprocessing

- Handling remaining missing Data: Imputation
- OneHotEncoding for Categorical Vairables
- Feature Selection based on Correlation Matrix
- Feature Selection based on Information Gain
- Feature Selection based on Automated Methods i.e. SelectKBest()

In [ ]:
```
# Dealing with missing values via imputation
# We went with median as it is a more robust to outliers choice and we di

from sklearn.impute import SimpleImputer

median_imp = SimpleImputer(strategy='median', add_indicator=False)
mode_imp = SimpleImputer(strategy='most_frequent', add_indicator=False)
```

In [ ]:
```
# One Hot Encoding of Categorical Variables
# For the purpose of the ML algorithms we are planning to use, we needed

from sklearn.preprocessing import OneHotEncoder

ohe = OneHotEncoder(sparse_output=False, drop='if_binary')
```

In [ ]:
```
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer

# Separating columns into numeric and categorical
# These will also be the columns that are the results of feature permutat

numeric_features = ["Age", "time_bw_schedule_appointment"]
categorical_features = ['SocialWelfare', 'Sex', 'Alcoholism', 'Hipertensi

# Creating transformers and encapsulating them in pipelines.
```

```
# 1. For handling the numerical features with only the imputer
# 2. For handling the categorical variables with imputer and the one hot

numeric_transformer = Pipeline(steps=[
    ('imputer', median_imp)
])

categorical_transformer = Pipeline(steps=[

    ('imputer', mode_imp),
    ('onehot', ohe)
])

# Applying transformers using ColumnTransformer

preprocessor = ColumnTransformer(
    transformers=[
        ('numeric', numeric_transformer, numeric_features),
        ('categorical', categorical_transformer, categorical_features)
    ])

# Applying the column transformer to the X_train to get rid of missing va

transformed_train = preprocessor.fit_transform(X_train)
```

In [ ]:

```
# Confirming we have no remaining missing values in X_train

columns_train = preprocessor.get_feature_names_out()
transformed_train = pd.DataFrame(transformed_train, columns=columns_train
transformed_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 61652 entries, 0 to 61651
Data columns (total 87 columns):
 #   Column                                         Non-Null Count  D
type
---  ------                                         --------------  -
----
 0   numeric__Age                                   61652 non-null  f
loat64
 1   numeric__time_bw_schedule_appointment          61652 non-null  f
loat64
 2   categorical__SocialWelfare_Yes                 61652 non-null  f
loat64
 3   categorical__Sex_M                             61652 non-null  f
loat64
 4   categorical__Alcoholism_Yes                    61652 non-null  f
loat64
 5   categorical__Hipertension_Yes                  61652 non-null  f
loat64
 6   categorical__Community_Aeroporto               61652 non-null  f
loat64
 7   categorical__Community_Andorinhas              61652 non-null  f
loat64
 8   categorical__Community_Antônio honório         61652 non-null  f
loat64
 9   categorical__Community_Ariovaldo favalessa     61652 non-null  f
loat64
 10  categorical__Community_Barro vermelho          61652 non-null  f
loat64
 11  categorical__Community_Bela vista              61652 non-null  f
loat64
 12  categorical__Community_Bento ferreira          61652 non-null  f
loat64
 13  categorical__Community_Boa vista               61652 non-null  f
loat64
 14  categorical__Community_Bonfim                  61652 non-null  f
loat64
 15  categorical__Community_Caratoíra               61652 non-null  f
loat64
 16  categorical__Community_Centro                  61652 non-null  f
loat64
 17  categorical__Community_Comdusa                 61652 non-null  f
loat64
 18  categorical__Community_Conquista               61652 non-null  f
loat64
 19  categorical__Community_Consolação              61652 non-null  f
loat64
 20  categorical__Community_Cruzamento              61652 non-null  f
loat64
 21  categorical__Community_Da penha                61652 non-null  f
loat64
 22  categorical__Community_De lourdes              61652 non-null  f
loat64
 23  categorical__Community_Do cabral               61652 non-null  f
loat64
 24  categorical__Community_Do moscoso              61652 non-null  f
loat64
 25  categorical__Community_Do quadro               61652 non-null  f
loat64
 26  categorical__Community_Enseada do suá          61652 non-null  f
```

```
loat64
 27  categorical__Community_Estrelinha                 61652 non-null  f
loat64
 28  categorical__Community_Fonte grande               61652 non-null  f
loat64
 29  categorical__Community_Forte são joão             61652 non-null  f
loat64
 30  categorical__Community_Fradinhos                  61652 non-null  f
loat64
 31  categorical__Community_Goiabeiras                 61652 non-null  f
loat64
 32  categorical__Community_Grande vitória             61652 non-null  f
loat64
 33  categorical__Community_Gurigica                   61652 non-null  f
loat64
 34  categorical__Community_Horto                      61652 non-null  f
loat64
 35  categorical__Community_Ilha das caieiras          61652 non-null  f
loat64
 36  categorical__Community_Ilha de santa maria        61652 non-null  f
loat64
 37  categorical__Community_Ilha do boi                61652 non-null  f
loat64
 38  categorical__Community_Ilha do frade              61652 non-null  f
loat64
 39  categorical__Community_Ilha do príncipe           61652 non-null  f
loat64
 40  categorical__Community_Ilhas oceânicas de trindade  61652 non-null  f
loat64
 41  categorical__Community_Inhanguetá                 61652 non-null  f
loat64
 42  categorical__Community_Itararé                    61652 non-null  f
loat64
 43  categorical__Community_Jabour                     61652 non-null  f
loat64
 44  categorical__Community_Jardim camburi             61652 non-null  f
loat64
 45  categorical__Community_Jardim da penha            61652 non-null  f
loat64
 46  categorical__Community_Jesus de nazareth          61652 non-null  f
loat64
 47  categorical__Community_Joana d´arc                61652 non-null  f
loat64
 48  categorical__Community_Jucutuquara               61652 non-null  f
loat64
 49  categorical__Community_Maria ortiz                61652 non-null  f
loat64
 50  categorical__Community_Maruípe                    61652 non-null  f
loat64
 51  categorical__Community_Mata da praia              61652 non-null  f
loat64
 52  categorical__Community_Monte belo                 61652 non-null  f
loat64
 53  categorical__Community_Morada de camburi          61652 non-null  f
loat64
 54  categorical__Community_Mário cypreste             61652 non-null  f
loat64
 55  categorical__Community_Nazareth                   61652 non-null  f
loat64
 56  categorical__Community_Nova palestina             61652 non-null  f
```

```
loat64
 57  categorical__Community_Parque industrial           61652 non-null  f
loat64
 58  categorical__Community_Parque moscoso              61652 non-null  f
loat64
 59  categorical__Community_Piedade                     61652 non-null  f
loat64
 60  categorical__Community_Pontal de camburi           61652 non-null  f
loat64
 61  categorical__Community_Praia do canto              61652 non-null  f
loat64
 62  categorical__Community_Praia do suá                61652 non-null  f
loat64
 63  categorical__Community_Redenção                    61652 non-null  f
loat64
 64  categorical__Community_República                   61652 non-null  f
loat64
 65  categorical__Community_Resistência                 61652 non-null  f
loat64
 66  categorical__Community_Romão                       61652 non-null  f
loat64
 67  categorical__Community_Santa cecília               61652 non-null  f
loat64
 68  categorical__Community_Santa clara                 61652 non-null  f
loat64
 69  categorical__Community_Santa helena                61652 non-null  f
loat64
 70  categorical__Community_Santa luíza                 61652 non-null  f
loat64
 71  categorical__Community_Santa lúcia                 61652 non-null  f
loat64
 72  categorical__Community_Santa martha                61652 non-null  f
loat64
 73  categorical__Community_Santa tereza                61652 non-null  f
loat64
 74  categorical__Community_Santo andré                 61652 non-null  f
loat64
 75  categorical__Community_Santo antônio               61652 non-null  f
loat64
 76  categorical__Community_Santos dumont               61652 non-null  f
loat64
 77  categorical__Community_Santos reis                 61652 non-null  f
loat64
 78  categorical__Community_Segurança do lar            61652 non-null  f
loat64
 79  categorical__Community_Solon borges               61652 non-null  f
loat64
 80  categorical__Community_São benedito               61652 non-null  f
loat64
 81  categorical__Community_São cristóvão              61652 non-null  f
loat64
 82  categorical__Community_São josé                   61652 non-null  f
loat64
 83  categorical__Community_São pedro                  61652 non-null  f
loat64
 84  categorical__Community_Tabuazeiro                 61652 non-null  f
loat64
 85  categorical__Community_Universitário              61652 non-null  f
loat64
 86  categorical__Community_Vila rubim                 61652 non-null  f
```

```
loat64
dtypes: float64(87)
memory usage: 40.9 MB
```

In [ ]:

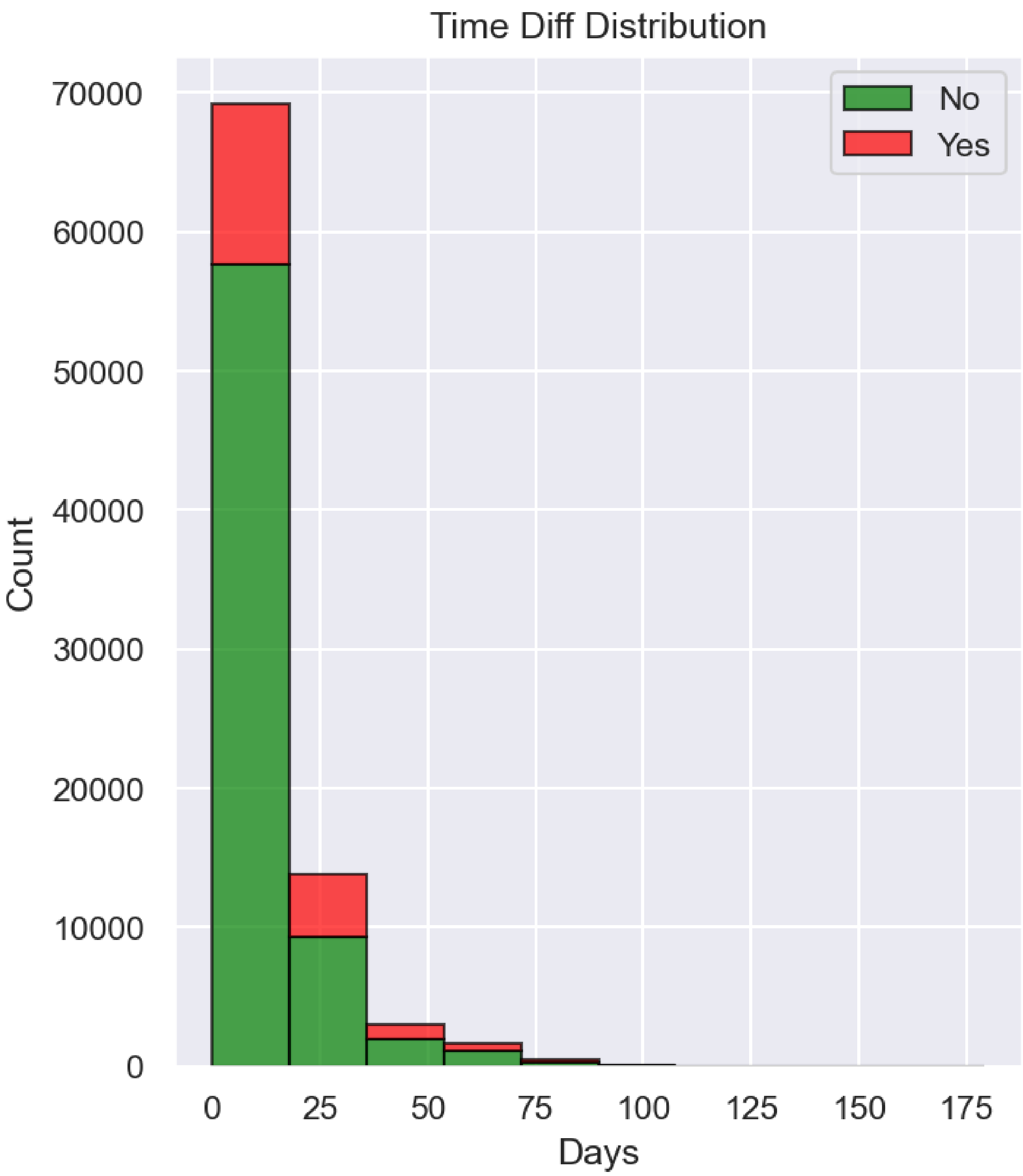
```
# Confirming we have no remaining missing values in X_test

transformed_test = preprocessor.fit_transform(X_test)

# Seeing the feature names in order to 'see beneath the hood'

columns_test = preprocessor.get_feature_names_out()
transformed_test = pd.DataFrame(transformed_test, columns= columns_test)
transformed_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26765 entries, 0 to 26764
Data columns (total 84 columns):
 #   Column                                   Non-Null Count  Dtype  
---  ------                                   --------------  -----  
 0   numeric__Age                             26765 non-null  float64
 1   numeric__time_bw_schedule_appointment    26765 non-null  float64
 2   categorical__SocialWelfare_Yes           26765 non-null  float64
 3   categorical__Sex_M                       26765 non-null  float64
 4   categorical__Alcoholism_Yes              26765 non-null  float64
 5   categorical__Hipertension_Yes            26765 non-null  float64
 6   categorical__Community_Aeroporto         26765 non-null  float64
 7   categorical__Community_Andorinhas        26765 non-null  float64
 8   categorical__Community_Antônio honório   26765 non-null  float64
 9   categorical__Community_Ariovaldo favalessa  26765 non-null  float64
 10  categorical__Community_Barro vermelho    26765 non-null  float64
 11  categorical__Community_Bela vista        26765 non-null  float64
 12  categorical__Community_Bento ferreira    26765 non-null  float64
 13  categorical__Community_Boa vista         26765 non-null  float64
 14  categorical__Community_Bonfim            26765 non-null  float64
 15  categorical__Community_Caratoíra         26765 non-null  float64
 16  categorical__Community_Centro            26765 non-null  float64
 17  categorical__Community_Comdusa           26765 non-null  float64
 18  categorical__Community_Conquista         26765 non-null  float64
 19  categorical__Community_Consolação        26765 non-null  float64
 20  categorical__Community_Cruzamento        26765 non-null  float64
 21  categorical__Community_Da penha          26765 non-null  float64
 22  categorical__Community_De lourdes        26765 non-null  float64
 23  categorical__Community_Do cabral         26765 non-null  float64
 24  categorical__Community_Do moscoso        26765 non-null  float64
 25  categorical__Community_Do quadro         26765 non-null  float64
 26  categorical__Community_Enseada do suá    26765 non-null  float64
 27  categorical__Community_Estrelinha        26765 non-null  float64
 28  categorical__Community_Fonte grande      26765 non-null  float64
 29  categorical__Community_Forte são joão    26765 non-null  float64
 30  categorical__Community_Fradinhos         26765 non-null  float64
 31  categorical__Community_Goiabeiras        26765 non-null  float64
 32  categorical__Community_Grande vitória    26765 non-null  float64
 33  categorical__Community_Gurigica          26765 non-null  float64
 34  categorical__Community_Horto             26765 non-null  float64
 35  categorical__Community_Ilha das caieiras  26765 non-null  float64
 36  categorical__Community_Ilha de santa maria  26765 non-null  float64
 37  categorical__Community_Ilha do boi       26765 non-null  float64
 38  categorical__Community_Ilha do príncipe  26765 non-null  float64
 39  categorical__Community_Inhanguetá        26765 non-null  float64
 40  categorical__Community_Itararé           26765 non-null  float64
 41  categorical__Community_Jabour            26765 non-null  float64
 42  categorical__Community_Jardim camburi    26765 non-null  float64
 43  categorical__Community_Jardim da penha   26765 non-null  float64
 44  categorical__Community_Jesus de nazareth  26765 non-null  float64
 45  categorical__Community_Joana d´arc       26765 non-null  float64
 46  categorical__Community_Jucutuquara       26765 non-null  float64
 47  categorical__Community_Maria ortiz       26765 non-null  float64
 48  categorical__Community_Maruípe           26765 non-null  float64
 49  categorical__Community_Mata da praia     26765 non-null  float64
 50  categorical__Community_Monte belo        26765 non-null  float64
 51  categorical__Community_Morada de camburi  26765 non-null  float64
 52  categorical__Community_Mário cypreste    26765 non-null  float64
 53  categorical__Community_Nazareth          26765 non-null  float64
 54  categorical__Community_Nova palestina    26765 non-null  float64
```

```
 55  categorical__Community_Parque moscoso      26765 non-null  float64
 56  categorical__Community_Piedade             26765 non-null  float64
 57  categorical__Community_Pontal de camburi   26765 non-null  float64
 58  categorical__Community_Praia do canto      26765 non-null  float64
 59  categorical__Community_Praia do suá        26765 non-null  float64
 60  categorical__Community_Redenção            26765 non-null  float64
 61  categorical__Community_República           26765 non-null  float64
 62  categorical__Community_Resistência         26765 non-null  float64
 63  categorical__Community_Romão               26765 non-null  float64
 64  categorical__Community_Santa cecília       26765 non-null  float64
 65  categorical__Community_Santa clara         26765 non-null  float64
 66  categorical__Community_Santa helena        26765 non-null  float64
 67  categorical__Community_Santa luíza         26765 non-null  float64
 68  categorical__Community_Santa lúcia         26765 non-null  float64
 69  categorical__Community_Santa martha        26765 non-null  float64
 70  categorical__Community_Santa tereza        26765 non-null  float64
 71  categorical__Community_Santo andré         26765 non-null  float64
 72  categorical__Community_Santo antônio       26765 non-null  float64
 73  categorical__Community_Santos dumont       26765 non-null  float64
 74  categorical__Community_Santos reis         26765 non-null  float64
 75  categorical__Community_Segurança do lar    26765 non-null  float64
 76  categorical__Community_Solon borges        26765 non-null  float64
 77  categorical__Community_São benedito        26765 non-null  float64
 78  categorical__Community_São cristóvão       26765 non-null  float64
 79  categorical__Community_São josé            26765 non-null  float64
 80  categorical__Community_São pedro           26765 non-null  float64
 81  categorical__Community_Tabuazeiro          26765 non-null  float64
 82  categorical__Community_Universitário       26765 non-null  float64
 83  categorical__Community_Vila rubim          26765 non-null  float64
dtypes: float64(84)
memory usage: 17.2 MB
```

In [ ]:

```
# EDA: Plotting histogram of split of time duration between scheduled and
# From a first glance it does not seem that obvious.

plt.hist([df_t[df_t['No-show'] == 'No']['time_bw_schedule_appointment'],
         bins=10, color=['green', 'red'], alpha=0.7, edgecolor='black', l

plt.title('Time Diff Distribution')
plt.xlabel('Days')
plt.ylabel('Count')
plt.legend()

plt.show()
```



7.2 Feature Selection

We would like to understand which features are meaningul in terms of information gain and predictive power while which ones simply add to the noise.

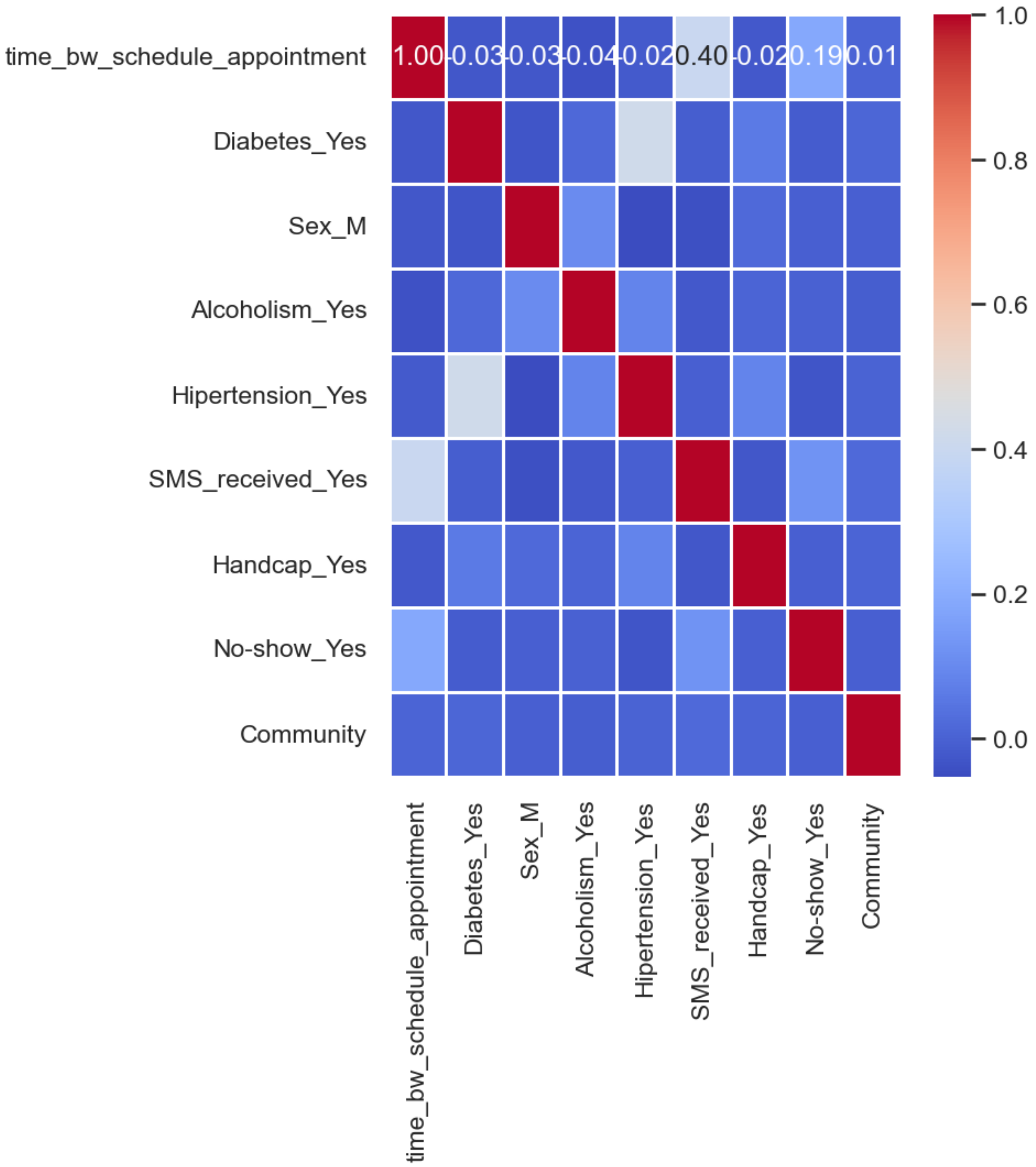
- Feature Selection based on Correlation Matrix
- Feature Selection based on Information Gain
- Feature Selection based on Automated Methods i.e. SelectKBest()

In [ ]:
```
## Concatenating X_train and y_train for feature selection

transformed_train = transformed_train.reset_index(drop=True)
y_train = y_train.reset_index(drop=True)

df_feature_selection = pd.concat([transformed_train, y_train], axis = 1)
```

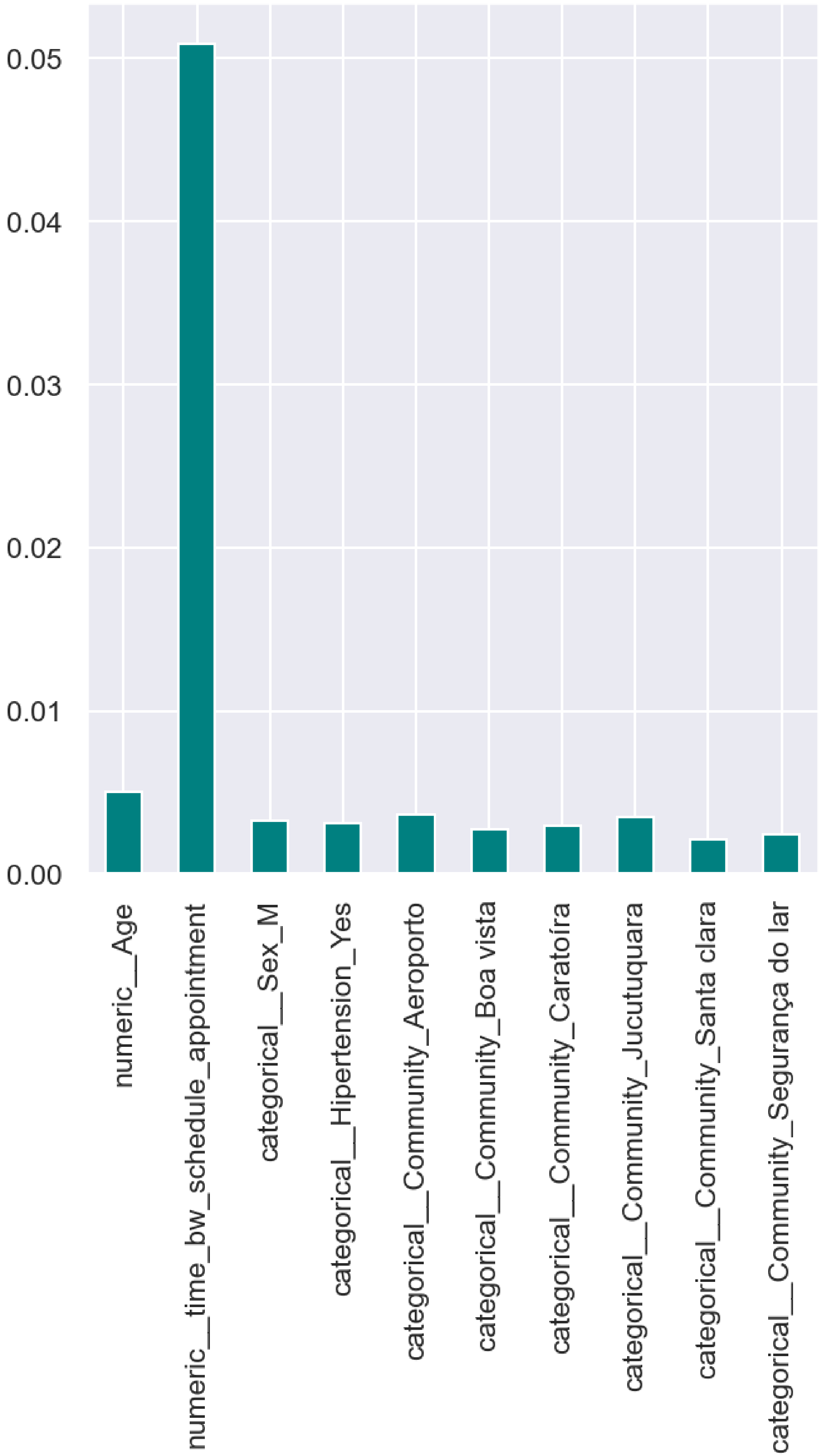
In [ ]:
```
# Encoding the no-show column to int to be able to use in feature selecti

df_t_encoded = pd.get_dummies(df_feature_selection, columns=['No-show'],
```

In [ ]:
```
# Feature Selection based on Correlation Matrix

from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()

df_sub = df_t[['Diabetes', 'Sex', 'Alcoholism', 'Hipertension', 'SMS_rece

df_t_encoded = pd.get_dummies(df_sub, columns=['Diabetes', 'Sex', 'Alcoho

df_t_encoded["Community"] = label_encoder.fit_transform(df_t["Community"]
correlation_matrix = df_t_encoded.corr()

sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", l
```

Out[ ]:
```
<Axes: >
```



In [ ]:
```
# Feature Selection based on Info Gain

from sklearn.feature_selection import mutual_info_classif

X = df_t_encoded.drop("No-show_Yes", axis=1)
y = df_t_encoded['No-show_Yes']

importances = mutual_info_classif(X, y)
feature_importances = pd.Series(importances, df_t_encoded.columns[0:len(d

# Filtering for the relatively more important ones after seeing a messy c

filtered_importances = feature_importances[feature_importances > 0.002]

filtered_importances.plot(kind="bar", color="teal")
plt.show()
```


0.05
0.04
0.03
0.02
0.01
0.00
numeric__Age
numeric__time_bw_schedule_appointment
categorical__Sex_M
categorical__Hipertension_Yes
categorical__Community_Aeroporto
categorical__Community_Boa vista
categorical__Community_Caratoíra
categorical__Community_Jucutuquara
categorical__Community_Santa clara
categorical__Community_Segurança do lar

In [ ]:
```
# Trying out an alternative feature selection method: KBest.

from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2, f_regression, f_classif
from numpy import array

# Creating training set and prediction target

X = df_t_encoded.drop("No-show_Yes", axis=1)
y = df_t_encoded['No-show_Yes']

# Performing feature selection
# We arbitrarily chose 4 as the number of features.

select = SelectKBest(score_func=f_classif, k=4)
select.fit_transform(X,y)

filter = select.get_support()
features = array(X.columns)

print(features[filter])
```

```
['numeric__Age' 'numeric__time_bw_schedule_appointment'
 'categorical__SocialWelfare_Yes' 'categorical__Hipertension_Yes']
```

The two methods presented age and time bw schedule appointment as the most important ones while there were conflicts in some categorical ones. Later we try a manual approach to find the combination of features which reaches a good level of generalization.

8. Model Building

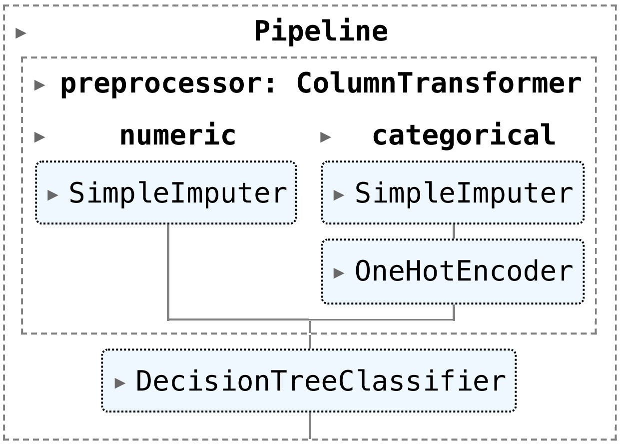
8.1 Decision Tree

We started with a simple decision tree as a common classification algorithm to see how a 'weak' classifier would work on the dataset. Turned out using this in the soft voting classifier brings the score down so we took it out of the soft voting classifier.

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.pipeline import Pipeline

clf = DecisionTreeClassifier()

# Creating pipeline for decision tree using the preprocessor from advance

pipe_dt = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', clf)]
)
```

In [ ]:
```
# Pipeline shape

pipe_dt
```

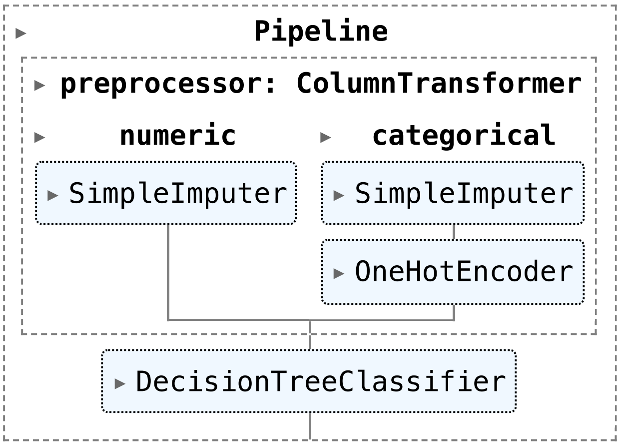
Out[ ]:



In [ ]:

```
# Training the model

pipe_dt.fit(X_train, y_train)
```

Out[ ]:

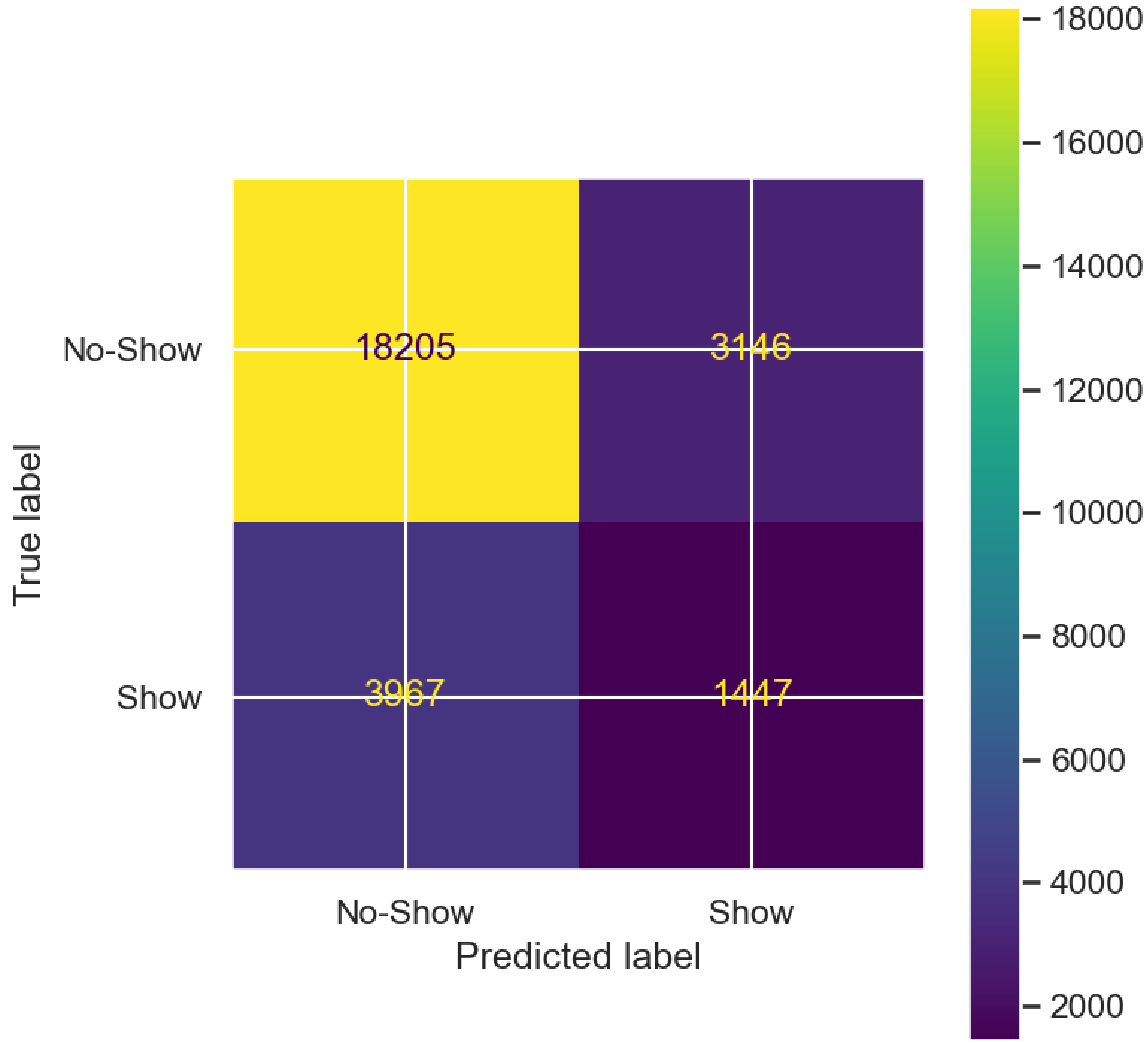


In [ ]:

```
# Making the predictions

y_pred = pipe_dt.predict(X_test)
```

In [ ]:

```
# Evaluating the model

import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

display_labels=['No-Show', 'Show']
cm = confusion_matrix(y_test, y_pred, labels=clf.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=display_labels)
disp.plot()
plt.show()
```



In [ ]:
```
# Print a classification report

from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

          No       0.82      0.85      0.84     21351
         Yes       0.32      0.27      0.29      5414

    accuracy                           0.73     26765
   macro avg       0.57      0.56      0.56     26765
weighted avg       0.72      0.73      0.73     26765
```

We see room for improvement considering the low f1 score for 'Yes' so build other models: XGBoost, Logistic Regression, and Random Forrest.

8.2 Random Forest

- Due to decision tree having a lower score, we decide to make a random forest as this is many decision trees combined together and voting on the most probably outcomes; therefore, increasing the predictive power.
- Essentially, bootstrapping!

In [ ]:

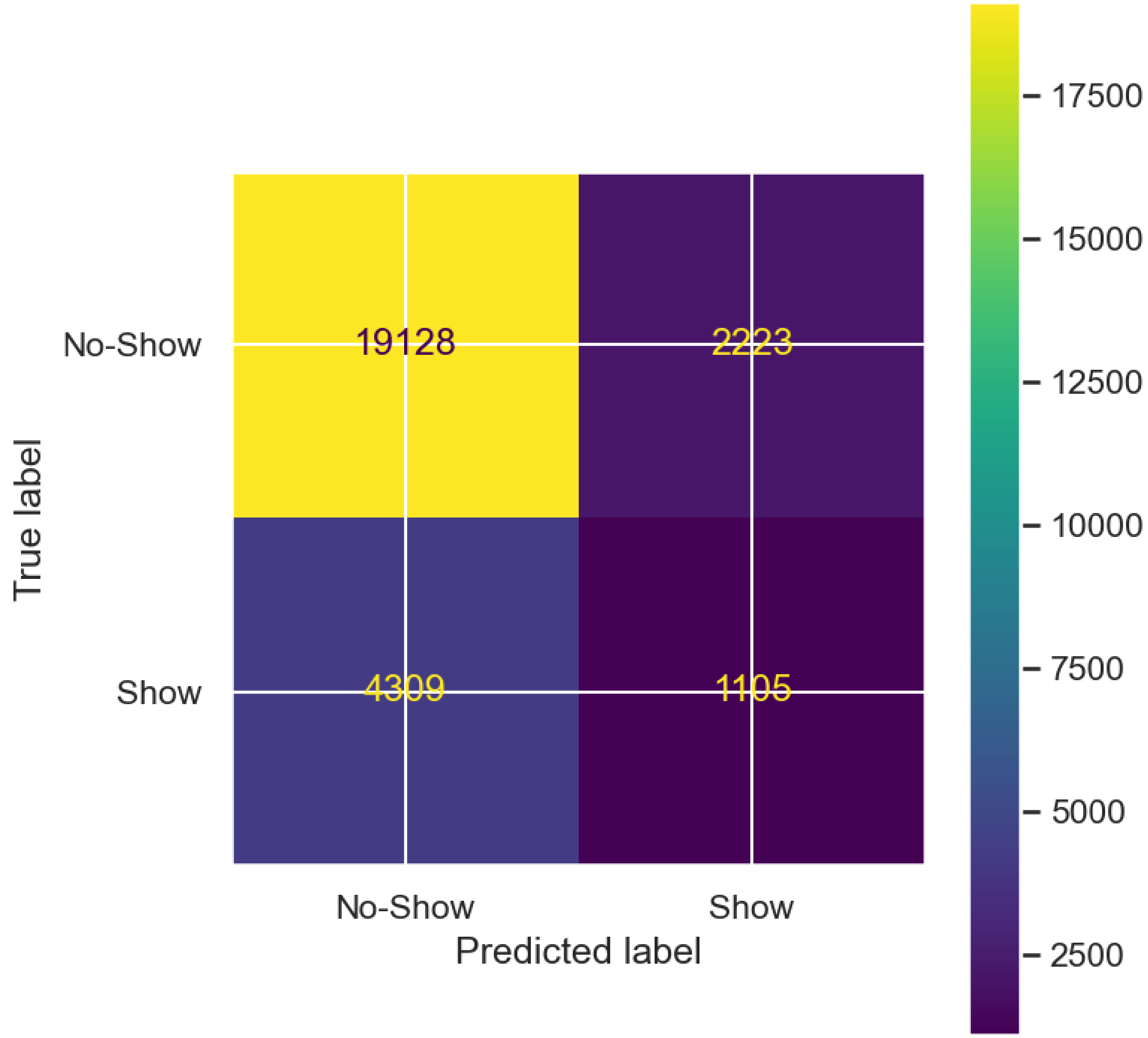
```python
from sklearn.ensemble import RandomForestClassifier

# Instantiating Random Forest

# 200 trees to be built with criteria on how to shape each tree and how m

rf = RandomForestClassifier(n_estimators=200, min_samples_split=2, min_sa

pipe_rf = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', rf)]
)

# Training the model

pipe_rf.fit(X_train, y_train)
```

Out[ ]:

```
▸ Pipeline
▸ preprocessor: ColumnTransformer
▸ numeric                ▸ categorical
▸ SimpleImputer          ▸ SimpleImputer
                         ▸ OneHotEncoder
▸ RandomForestClassifier
```

In [ ]:

```python
# Making predicitions

preds_rf = pipe_rf.predict(X_test)
preds_rf
display_labels=['No-Show', 'Show']
cm = confusion_matrix(y_test, preds_rf, labels=pipe_rf.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                               display_labels=display_labels)
disp.plot()
plt.show()
from sklearn.metrics import classification_report
print(classification_report(y_test, preds_rf))
```




```
              precision    recall  f1-score   support

           0       0.82      0.90      0.85     21351
           1       0.33      0.20      0.25      5414

    accuracy                           0.76     26765
   macro avg       0.57      0.55      0.55     26765
weighted avg       0.72      0.76      0.73     26765
```

In [ ]:
```
# Ensuring y train and y test are encoded binary

mapping = {'No': 0, 'Yes': 1}
y_train = y_train.replace(mapping)
y_test = y_test.replace(mapping)
```

In [ ]:
```
# We are removing these columns as they were not found to be present with
# There could potentially be more robust ways of doing this; however, con

transformed_train = transformed_train.drop(columns=['categorical__Communi
```

8.2.1 Random Search

- Random Search allows us to specify a set of parameters for the algorithm to sift through and choose the best combination.

- It helps us more than GridSearch as we expect lower marginal value from GridSearch compared to the computation time.
- We focused on not having an overfitted model by optimizing for max_depth, min_samples_split, and min_samples_left.
- On the other hand we also did not want an underfitted model which cannot pick up trends sufficiently so we used n_estimators.

In [ ]:
```
# Optimizing HyperParameters

from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score

param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2']
}

# Creating a RandomizedSearchCV object

random_search = RandomizedSearchCV(rf, param_distributions=param_grid, n_

# Fitting the RandomizedSearchCV object on training

random_search.fit(transformed_train, y_train)

# Retrieving the best model and hyperparameters

best_model = random_search.best_estimator_
best_params = random_search.best_params_

# Evaluating the best model on the test set

y_pred = best_model.predict(transformed_test)

accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

print("Best Hyperparameters:", best_params)
print(f"Best Model Accuracy on Test Set: {accuracy:.4f}")
print(f"Best Model F1 on Test Set: {f1:.4f}")
print(f"Best Model Recall on Test Set: {recall:.4f}")
```

```
Best Hyperparameters: {'n_estimators': 200, 'min_samples_split': 2, 'min_s
amples_leaf': 1, 'max_features': 'log2', 'max_depth': None}
Best Model Accuracy on Test Set: 0.7585
Best Model F1 on Test Set: 0.2553
Best Model Recall on Test Set: 0.2047
```

8.2.2. Halving Grid Search

- We wanted to compare the hyperparameter tuning approach to some other approach.

- Used Halving Grid Search as it is less computationally expensive as Grid Search but also does work on promising hyperparameter configurations.

In [ ]:
```
# enable_halving_search_cv is implicitly needed but not explicitly called

from sklearn.experimental import enable_halving_search_cv
from sklearn.model_selection import HalvingGridSearchCV

# Defining the hyperparameter grid

param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2']

}

# Creating a HalvingGridSearchCV object (Hyperband with grid search)

hyperband_search = HalvingGridSearchCV(rf, param_grid, factor=3, cv=3, ra

# Fitting the Hyperband search on the training data

hyperband_search.fit(transformed_train, y_train)

best_rf_model = hyperband_search.best_estimator_
best_params = hyperband_search.best_params_

y_pred = best_rf_model.predict(transformed_test)

f1 = f1_score(y_test, y_pred)

print(f"Best Model F1 on Test Set: {f1:.4f}")
print("Best Hyperparameters:", best_params)
```

```
Best Model F1 on Test Set: 0.2594
Best Hyperparameters: {'max_depth': None, 'max_features': 'log2', 'min_sam
ples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}
```

We see that the accuracy and other metrics are almost similar - even after the hyper parameter tuning.

8.3 XGBoost

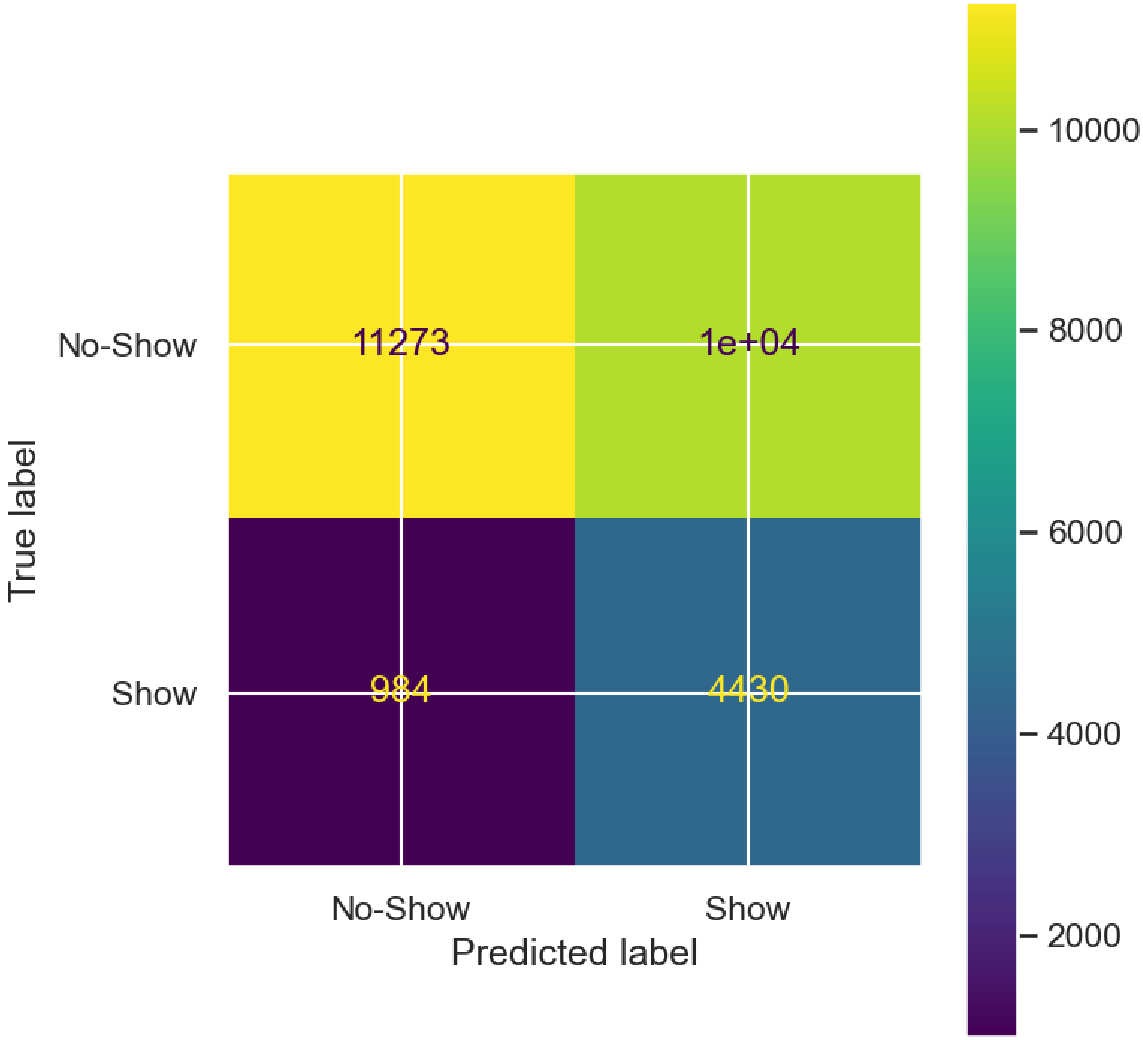
- To further test out if we could improve the prediction power, we tried more model models.
- We used XGBoost as this is widely considered one of the best performing classifiers.
- Although the algorithm does also create decision trees, the key difference is the use of a boosting method and not a simple bagging one - they pick up on errors of each tree.

In [ ]:
```
from xgboost import XGBClassifier
```

```
# Giving more weight to the minority class
# XGBoost is sensitive to imbalanced classes

num_positive_instances = np.sum(y_train == 1)
num_negative_instances = np.sum(y_train == 0)

imbalance_ratio = num_negative_instances / num_positive_instances

# Using a binary classifier objective, inputting the imbalance ratio we c
# We use binary:logistic as objective as we would like the probabilities

xgb = XGBClassifier(objective='binary:logistic', scale_pos_weight = imbal

pipe_xgb = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', xgb)]
)

pipe_xgb.fit(X_train, y_train)
preds_xgb = pipe_xgb.predict(X_test)
```

In [ ]:

```
display_labels=['No-Show', 'Show']
cm = confusion_matrix(y_test, preds_xgb, labels=pipe_xgb.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=display_labels)
disp.plot()
plt.show()
```



In [ ]:
```
# Classification Report

from sklearn.metrics import classification_report
print(classification_report(y_test, preds_xgb))
```

```
              precision    recall  f1-score   support

           0       0.92      0.53      0.67     21351
           1       0.31      0.82      0.44      5414

    accuracy                           0.59     26765
   macro avg       0.61      0.67      0.56     26765
weighted avg       0.80      0.59      0.63     26765
```

8.3.1 Random Search

Parameters:

- n_estimators, max_depth, learning_rate: to ensure a balance between under and over fitting
- gamme: the regularization parameter (since XGBoost supports built-in regularization)
- scale_pos_weight: playing around with different weights of the class imbalances

In [ ]:
```
# Optimizing HyperParameters

from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score

param_dist = {
    'n_estimators': [50, 100, 200],
    'max_depth': [3, 5, 7, 10],
    'learning_rate': [0.01, 0.1, 0.2, 0.3],
    'subsample': [0.8, 0.9, 1.0],
    'colsample_bytree': [0.8, 0.9, 1.0],
    'gamma': [0, 1, 5],
    'scale_pos_weight': [2, 4, 6, 8]
}

# Creating a RandomizedSearchCV object

random_search = RandomizedSearchCV(xgb, param_distributions=param_dist, n

# Fitting the RandomizedSearchCV object on training set

random_search.fit(transformed_train, y_train)

# Retrieving the best model and hyperparameters

best_model = random_search.best_estimator_
best_params = random_search.best_params_

# Evaluating the best model on the test set

y_pred = best_model.predict(transformed_test)
y_pred
```

```
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

print("Best Hyperparameters:", best_params)
print(f"Best Model Accuracy on Test Set: {accuracy:.4f}")
print(f"Best Model F1 on Test Set: {f1:.4f}")
print(f"Best Model Recall on Test Set: {recall:.4f}")
```

```
Best Hyperparameters: {'subsample': 0.9, 'scale_pos_weight': 4, 'n_estimat
ors': 200, 'max_depth': 3, 'learning_rate': 0.2, 'gamma': 0, 'colsample_by
tree': 1.0}
Best Model Accuracy on Test Set: 0.5889
Best Model F1 on Test Set: 0.4462
Best Model Recall on Test Set: 0.8188
```

8.3.2. Halving Random Search

- Following a similar approach as Random Forest, we run Halving Random Search to see if we can find better a performing combination of hyperparameters.
- Halving Grid Search was too computationally expensive so a Halving Random Search was preferred.

In [ ]:
```
from sklearn.model_selection import HalvingRandomSearchCV

# Defining the hyperparameter grid

param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [3, 5, 7, 10],
    'learning_rate': [0.01, 0.1, 0.2, 0.3],
    'subsample': [0.8, 0.9, 1.0],
    'colsample_bytree': [0.8, 0.9, 1.0],
    'gamma': [0, 1, 5],
    'scale_pos_weight': [2, 4, 6, 8]

}

# Createing a HalvingGridSearchCV object (Hyperband with grid search)

hyperband_search = HalvingRandomSearchCV(xgb, param_grid, factor=3, cv=3,

# Fitting the Hyperband search on the training data

hyperband_search.fit(transformed_train, y_train)


best_rf_model = hyperband_search.best_estimator_
best_params = hyperband_search.best_params_

y_pred = best_rf_model.predict(transformed_test)

f1 = f1_score(y_test, y_pred)

print(f"Best Model F1 on Test Set: {f1:.4f}")
print("Best Hyperparameters:", best_params)
```

```
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/metrics/_classification.py:1760: UndefinedMetricWarning: F-s
core is ill-defined and being set to 0.0 due to no true nor predicted samp
les. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, "true nor predicted", "F-score is", len(true_sum))
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/metrics/_classification.py:1760: UndefinedMetricWarning: F-s
core is ill-defined and being set to 0.0 due to no true nor predicted samp
les. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, "true nor predicted", "F-score is", len(true_sum))
Best Model F1 on Test Set: 0.4362
Best Hyperparameters: {'subsample': 1.0, 'scale_pos_weight': 6, 'n_estimat
ors': 200, 'max_depth': 10, 'learning_rate': 0.1, 'gamma': 1, 'colsample_b
ytree': 0.8}
```

8.4 Logistic Regression

- We wanted to add variety to our approaches. So far we only had tree based models.
- Also, its a model that focuses on class probabilities and therefore would also help us in the soft voting classifier.

In [ ]:
```python
from sklearn.linear_model import LogisticRegression

# Instantiating Logistic Regression

# Parameter choice:
# Penalty term adds a control to over fitting. We use L2 as it is more st

lgr = LogisticRegression(class_weight = 'balanced', solver = 'liblinear',

pipe_lgr = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', lgr)]
)

pipe_lgr.fit(X_train, y_train)

preds_lgr = pipe_lgr.predict(X_test)
preds_lgr
```

Out[ ]:
```
array([1, 1, 1, ..., 1, 0, 0])
```

In [ ]:
```python
# Classification Report

from sklearn.metrics import classification_report
print(classification_report(y_test, preds_lgr))
```

```
              precision    recall  f1-score   support

           0       0.86      0.73      0.79     21351
           1       0.33      0.53      0.41      5414

    accuracy                           0.69     26765
   macro avg       0.60      0.63      0.60     26765
weighted avg       0.75      0.69      0.71     26765
```

8.4.1 Random Search

In [ ]:
```
# Optimizing HyperParameters

from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score

param_dist = {
    'penalty': ['l1', 'l2'],
    'C': [0.001, 0.01, 0.1, 1, 10, 100],
    'fit_intercept': [True, False],
    'solver': ['liblinear', 'saga', 'lbfgs'],
    'class_weight': [None, 'balanced', {0: 1, 1: 2}, {0: 1, 1: 4}]
}
# Creating a RandomizedSearchCV object

random_search = RandomizedSearchCV(lgr, param_distributions=param_dist, n

# Fitting the RandomizedSearchCV object on train

random_search.fit(transformed_train, y_train)

# Retrieving the best model and hyperparameters

best_model = random_search.best_estimator_
best_params = random_search.best_params_

# Evaluating the best model on the test set

y_pred = best_model.predict(transformed_test)
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

print("Best Hyperparameters:", best_params)
print(f"Best Model Accuracy on Test Set: {accuracy:.4f}")
print(f"Best Model F1 on Test Set: {f1:.4f}")
print(f"Best Model Recall on Test Set: {recall:.4f}")
```

```
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
```

```
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
```

```
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
```

```
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/model_selection/_validation.py:425: FitFailedWarning:
100 fits failed out of a total of 150.
The score on these train-test partitions for these parameters will be set
to nan.
If these failures are not expected, you can try to debug them by setting e
rror_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------
------
40 fits failed with the following error:
Traceback (most recent call last):
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/
site-packages/sklearn/model_selection/_validation.py", line 729, in _fit_a
nd_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/
site-packages/sklearn/base.py", line 1152, in wrapper
    return fit_method(estimator, *args, **kwargs)
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/
site-packages/sklearn/linear_model/_logistic.py", line 1169, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/
site-packages/sklearn/linear_model/_logistic.py", line 61, in _check_solve
r
    raise ValueError(
ValueError: Solver saga supports only dual=False, got dual=True

--------------------------------------------------------------------------
------
25 fits failed with the following error:
Traceback (most recent call last):
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/
site-packages/sklearn/model_selection/_validation.py", line 729, in _fit_a
nd_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/
site-packages/sklearn/base.py", line 1152, in wrapper
    return fit_method(estimator, *args, **kwargs)
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/
site-packages/sklearn/linear_model/_logistic.py", line 1228, in fit
    self.coef_, self.intercept_, self.n_iter_ = _fit_liblinear(
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/
site-packages/sklearn/svm/_base.py", line 1229, in _fit_liblinear
    solver_type = _get_liblinear_solver_type(multi_class, penalty, loss, d
ual)
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/
site-packages/sklearn/svm/_base.py", line 1060, in _get_liblinear_solver_t
ype
    raise ValueError(
ValueError: Unsupported set of arguments: The combination of penalty='l1'
and loss='logistic_regression' are not supported when dual=True, Parameter
s: penalty='l1', loss='logistic_regression', dual=True

--------------------------------------------------------------------------
------
15 fits failed with the following error:
Traceback (most recent call last):
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/
```

```
site-packages/sklearn/model_selection/_validation.py", line 729, in _fit_a
nd_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/
site-packages/sklearn/base.py", line 1152, in wrapper
    return fit_method(estimator, *args, **kwargs)
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/
site-packages/sklearn/linear_model/_logistic.py", line 1169, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/
site-packages/sklearn/linear_model/_logistic.py", line 61, in _check_solve
r
    raise ValueError(
ValueError: Solver lbfgs supports only dual=False, got dual=True

--------------------------------------------------------------------------
------
20 fits failed with the following error:
Traceback (most recent call last):
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/
site-packages/sklearn/model_selection/_validation.py", line 729, in _fit_a
nd_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/
site-packages/sklearn/base.py", line 1152, in wrapper
    return fit_method(estimator, *args, **kwargs)
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/
site-packages/sklearn/linear_model/_logistic.py", line 1169, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/
site-packages/sklearn/linear_model/_logistic.py", line 56, in _check_solve
r
    raise ValueError(
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 pe
nalty.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/model_selection/_search.py:979: UserWarning: One or more of
the test scores are non-finite: [0.40416139        nan        nan        n
an        nan 0.10042915
        nan        nan        nan 0.13987686        nan 0.25605375
        nan 0.18970113        nan        nan        nan        nan
 0.00967246 0.19772576 0.02536691        nan        nan        nan
        nan        nan        nan 0.10314318 0.32965624        nan]
  warnings.warn(
Best Hyperparameters: {'solver': 'liblinear', 'penalty': 'l2', 'fit_interc
ept': False, 'class_weight': {0: 1, 1: 4}, 'C': 0.001}
Best Model Accuracy on Test Set: 0.6655
Best Model F1 on Test Set: 0.4086
Best Model Recall on Test Set: 0.5713
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pac
kages/sklearn/svm/_base.py:1250: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
```

8.4.2 Halving Grid Search

We comment this part out since halving search did not provide better results than random search. Hence, there is no need to spend computational resources here.

In [ ]:
```
# from sklearn.experimental import enable_halving_search_cv
# from sklearn.model_selection import HalvingGridSearchCV
# from sklearn.datasets import make_classification


# # Define the hyperparameter grid

# param_grid = {
#     'penalty': ['l1', 'l2'],
#     'C': [0.001, 0.01, 0.1, 1, 10, 100],
#     'fit_intercept': [True, False],
#     'solver': ['liblinear', 'saga']

# }

# # Create a HalvingGridSearchCV object (Hyperband with grid search)
# hyperband_search = HalvingGridSearchCV(lgr1, param_grid, factor=3, cv=3

# # Fit the Hyperband search on the training data

# hyperband_search.fit(transformed_train, y_train)

# best_rf_model = hyperband_search.best_estimator_
# best_params = hyperband_search.best_params_

# y_pred = best_rf_model.predict(transformed_test)

# f1 = f1_score(y_test, y_pred)
# print(f"Best Model F1 on Test Set: {f1:.4f}")
# print("Best Hyperparameters:", best_params)
```

8.5 Voting Classifier

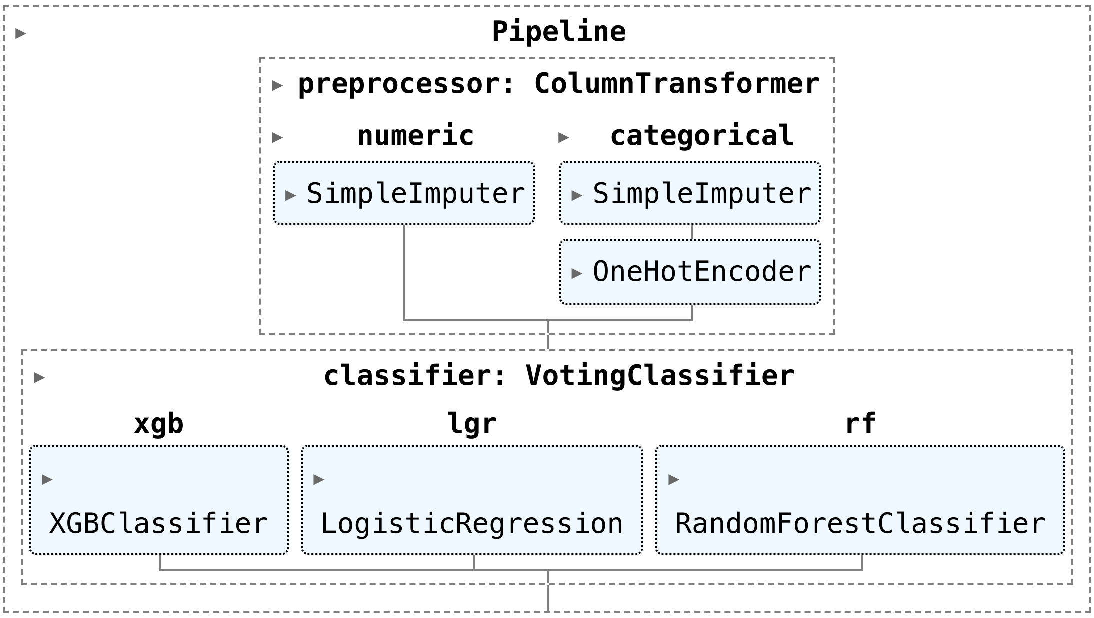
- Ensemble combines the power of all, so this was tried out and did prove to have the best results.
- We took the decision tree out since for voting classifier as it is not recommended to include algorithms with similar approaches. Therefore, we took the decision tree out.

In [ ]:
```
from sklearn.ensemble import VotingClassifier

# We observed after some hit and trial that these weights were providing

weights = [2, 2, 1]

## Compare both the soft and hard voting classifier.

svc = VotingClassifier(estimators=[('xgb', xgb), ('lgr', lgr), ('rf', rf)
hvc = VotingClassifier(estimators=[('xgb', xgb), ('lgr', lgr), ('rf', rf)

pipe_svc = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', svc)]
```

```
)

pipe_hvc = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', hvc)]
)
```

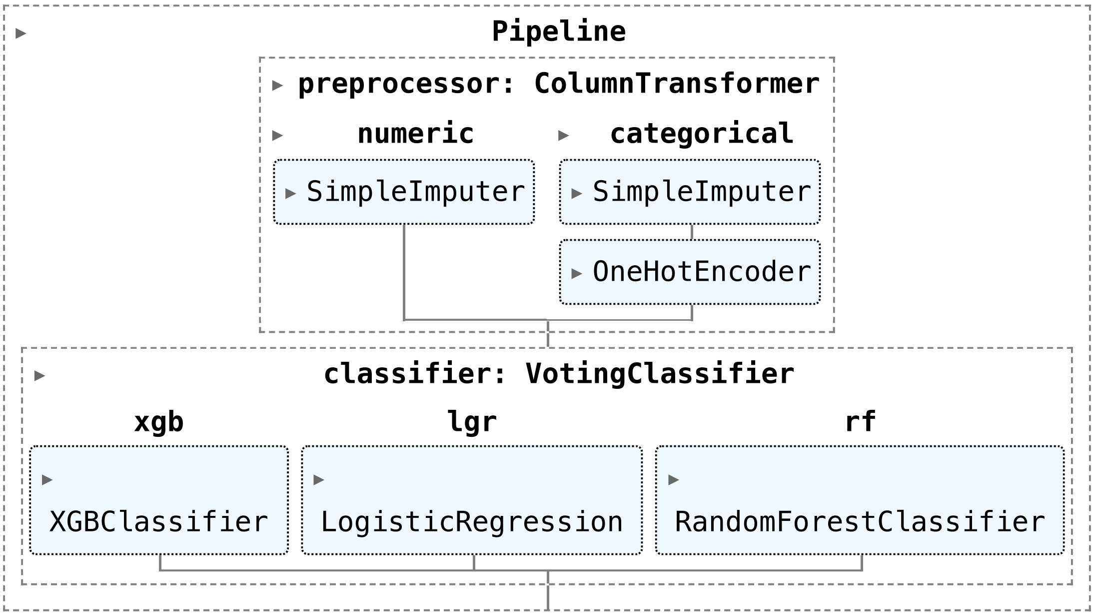
In [ ]:
```
pipe_svc.fit(X_train, y_train)
```

Out[ ]:



In [ ]:
```
pipe_hvc.fit(X_train, y_train)
```

Out[ ]:



In [ ]:
```
# Evaluating the classifiers

from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score

hard_voting_pred = pipe_hvc.predict(X_test)
soft_voting_pred = pipe_svc.predict(X_test)

print("F1 with Hard Voting:", f1_score(y_test, hard_voting_pred))
```

```
print("F1 with Soft Voting:", f1_score(y_test, soft_voting_pred))
print("Precision with Hard Voting:", precision_score(y_test, hard_voting_
print("Precision with Soft Voting:", precision_score(y_test, soft_voting_
print("Recall with Hard Voting:", recall_score(y_test, hard_voting_pred))
print("Recall with Soft Voting:", recall_score(y_test, soft_voting_pred))
```

```
F1 with Hard Voting: 0.4170117201207103
F1 with Soft Voting: 0.4210830324909747
Precision with Hard Voting: 0.33627617430673457
Precision with Soft Voting: 0.3456614509246088
Recall with Hard Voting: 0.5487624676763946
Recall with Soft Voting: 0.5386036202438124
```

In [ ]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, soft_voting_pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.74      0.80     21351
           1       0.35      0.54      0.42      5414

    accuracy                           0.70     26765
   macro avg       0.60      0.64      0.61     26765
weighted avg       0.76      0.70      0.72     26765
```

Soft Voting and Hard Voting Classifier show similar results on the training split, so we try both of them on the test.csv file. Results showed that we soft voting classifier outperformed the hard voting.

9. Model Evaluation

9.1 Feature Permutation

We are attempting a more advanced way of understanding how different features contribute to the predictions.

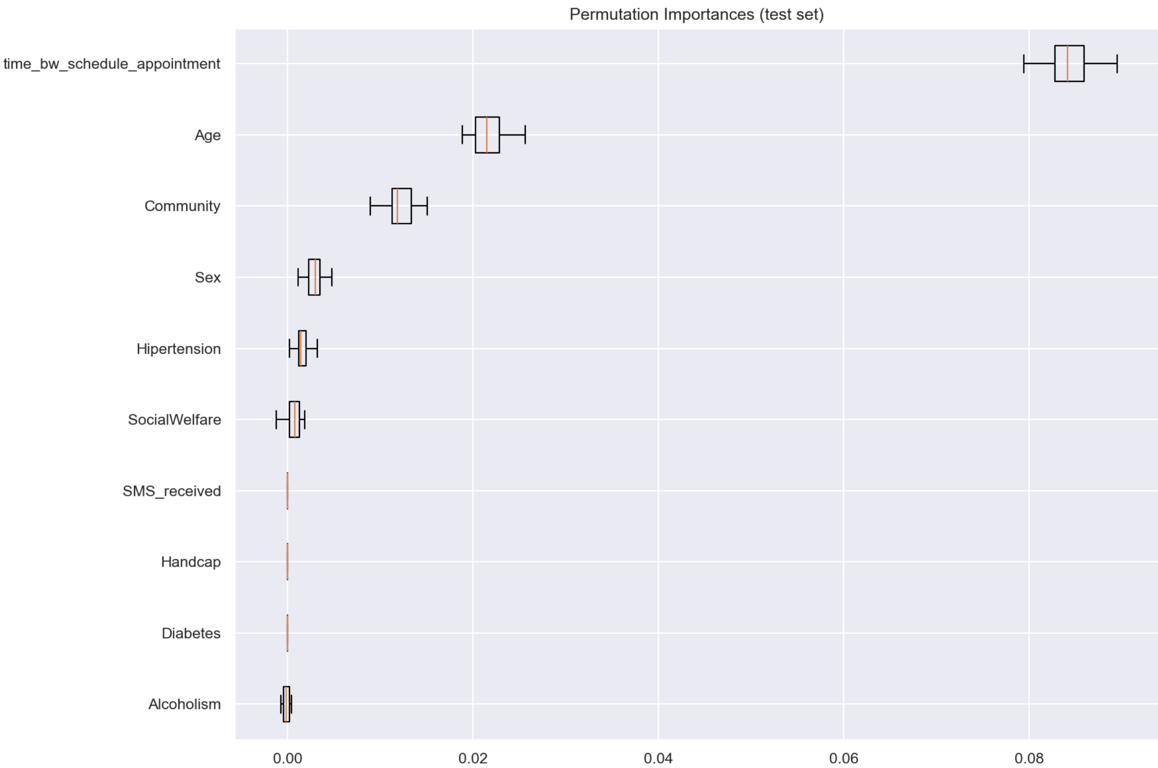
In [ ]:
```
from sklearn.metrics import accuracy_score

# Using accuracy as a simple to understand measure

y_pred_original = pipe_svc.predict(X_test)
accuracy_original = accuracy_score(y_test, y_pred_original)
print(f"Accuracy on original data: {accuracy_original:.4f}")
```

```
Accuracy on original data: 0.7004
```

In [ ]:
```
from sklearn.inspection import permutation_importance

# Permutation importance will shuffle the values of the chosen columns an
# decrease in the accuracy of the model, then the feature held importance

perm_result = permutation_importance(pipe_svc, X_test, y_test, n_repeats=
```

In [ ]:
```
import matplotlib.pyplot as plt
import numpy as np

# Sorting the features by importance
```

```
sorted_idx = perm_result.importances_mean.argsort()

# Plotting to visualise more clearly

plt.figure(figsize=(12, 8))
plt.boxplot(perm_result.importances[sorted_idx].T, vert=False, labels=np.
plt.title("Permutation Importances (test set)")
plt.tight_layout()
plt.show()
```



This resonates well with what we saw in feature selection efforts before. We use this to rethink our approach to considering columns such as SMS_received, handcap, diabetes, and alcoholism.

9.2 Preprocessing test.cv

We apply the same basic transformations that we did to our training data set.

In [ ]:
```
# Importing test.csv

df_test = pd.read_csv('/Users/muhammadraza/Documents/GitHub/BIPM/Data Sci

# Keeping original df_test intact:

df_test_t = df_test[:]
```

In [ ]:
```
## Data Type Overview

df_test_t.dtypes
```

Out[ ]:
```
PatientId          float64
AppointmentID        int64
Sex                 object
ScheduledDate       object
AppointmentDate     object
Age                float64
Community           object
SocialWelfare       object
Hipertension        object
Diabetes            object
Alcoholism          object
Handcap             object
SMS_received        object
dtype: object
```

In [ ]:
```
# Getting the dates in the right format

from datetime import datetime

df_test_t['AppointmentDate'] = df_test_t['AppointmentDate'].apply(lambda
df_test_t['ScheduledDate'] = df_test_t['ScheduledDate'].apply(lambda x: d
```

In [ ]:
```
# Replacing handcap numerical values to yes (same approach as in training

import numpy as np

df_test_t.loc[df_test_t['Handcap'].isin(['2', '3', '4']), 'Handcap'] = 'y
df_test_t.info()
occ_test = df_test_t.groupby('Handcap').size().reset_index()
print(occ_test)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22106 entries, 0 to 22105
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   PatientId        22106 non-null  float64
 1   AppointmentID    22106 non-null  int64
 2   Sex              22106 non-null  object
 3   ScheduledDate    22106 non-null  object
 4   AppointmentDate  22106 non-null  object
 5   Age              19955 non-null  float64
 6   Community        19461 non-null  object
 7   SocialWelfare    19043 non-null  object
 8   Hipertension     20089 non-null  object
 9   Diabetes         22106 non-null  object
 10  Alcoholism       18371 non-null  object
 11  Handcap          22106 non-null  object
 12  SMS_received     22106 non-null  object
dtypes: float64(2), int64(1), object(10)
memory usage: 2.2+ MB
  Handcap      0
0      no  21660
1     yes    446
```

In [ ]:
```
# Since we still see some missing data, we extrapolating missing data as

missing_columns = ['Age', 'Community', 'SocialWelfare', 'Hipertension', '
```

```
for column in missing_columns:
    df_test_t[column] = df_test_t.groupby('PatientId')[column].transform(
```

```
/var/folders/1z/kkxxkq_90qldq3ngyx1pj0fr0000gn/T/ipykernel_42276/73403312.
py:6: FutureWarning: Series.fillna with 'method' is deprecated and will ra
ise in a future version. Use obj.ffill() or obj.bfill() instead.
  df_test_t[column] = df_test_t.groupby('PatientId')[column].transform(lam
bda x: x.fillna(method='ffill').fillna(method='bfill'))
```

In [ ]:

```
# Adding New Feature: Time between ScheduledDate and AppointmentDate

df_test_t['time_bw_schedule_appointment'] = df_test_t['AppointmentDate']

## Convert to float (days)

df_test_t['time_bw_schedule_appointment'] = df_test_t['time_bw_schedule_a
df_test_t['time_bw_schedule_appointment'] = df_test_t['time_bw_schedule_a
```

In [ ]:

```
# Capitalising yes/no so they can be converted to binary column

df_test_t = df_test_t.applymap(lambda x: x.capitalize() if isinstance(x,
```

```
/var/folders/1z/kkxxkq_90qldq3ngyx1pj0fr0000gn/T/ipykernel_42276/47077837
8.py:3: FutureWarning: DataFrame.applymap has been deprecated. Use DataFra
me.map instead.
  df_test_t = df_test_t.applymap(lambda x: x.capitalize() if isinstance(x,
str) else x)
```

In [ ]:

```
# Making the columns for train and test equal by removing the columns we

X_df_test = df_test_t.drop(columns=['PatientId', 'AppointmentID', 'Schedu
```

9.2 Model Evaluation

Running the soft voting classifier on the test.csv file to see the performance on never seen before data.

In [ ]:

```
# Using the soft voting classifier pipe on the test dataset to make predi

y_pred_test = pipe_svc.predict(X_df_test)
```

In [ ]:

```
# Appending the results to the dataframe

df_test_t["No-show"] = y_pred_test
```

In [ ]:

```
# Reverse mapping of No-show column to Yes/No as is the requirement of th

mapping = {0: 'No', 1: 'Yes'}

# Consolidating into one file

df_test_t["No-show"] = df_test_t["No-show"].replace(mapping)
Submission = df_test_t[["AppointmentID", "No-show"]]

# Ensuring the final result is as expected

Submission.head()
```

Out[ ]:

	AppointmentID	No-show
0	5620835	Yes
1	5741692	No
2	5673005	No
3	5579701	No
4	5652332	No

In [ ]:

```
# Saving as csv in local directory

filepath = '/Users/muhammadraza/Documents/GitHub/BIPM/Data Science/Projec
Submission.to_csv(filepath, index=False)
```

10. Business Recommendations

How our model would help the business?

- The hospital could use our predictions to make a decision on whether to remind the people to come to the appointment or allocate the slot to other patients.

Are both false positives and false negatives equally important?

- It would depend on what the hospital would want to optimise for and what their resource utilisation is like. If the current resources are being under utilized, then it might make sense to have some overbookings and look at the false positives as it expects some patients to not show up but they do.
- If the hospital resources are already over utilised, then we would want to look at the false negatives more as they tell us which ones actually show up from the ones not expecting to show up. For the hospital, this would mean that they want to reduce the number of people who show up but expected not to show up. Thus freeing their resources and accurately being able to allocate resources elsewhere.