Machine Learning in Python - Group Project 1

Due Friday, March 10th by 16.00 pm.

Luke Appleby, Josep Reyero, Tom Birkbeck, Adam Dunajski

General Setup

In [5]:
```
# Add any additional libraries or submodules below

# Data libraries
import numpy as np
import pandas as pd

# Plotting libraries
import matplotlib.pyplot as plt
import seaborn as sns

# Plotting defaults
plt.rcParams['figure.figsize'] = (8,5)
plt.rcParams['figure.dpi'] = 80

# sklearn modules that are necessary
import sklearn
```

In [2]:
```
# Load data
data = pd.read_csv("the_office.csv")
data_explore = data.copy()
```

In [3]:
```
with open('MLP_cold_opens.txt', 'r') as file:
    new_data = file.read().replace('\n', '')

x = new_data.split(")")
hots = []
for i in range(len(x)):
    if x[i][0:4] == 'None':
        hots.append(x[i][7:-1])
    elif x[i][0:8] == 'Season 1' and x[i][8:12] == 'None':
        hots.append(x[i][14:])
    elif x[i][0:6] == 'Season' and x[i][8:12] == 'None':
        hots.append(x[i][15:-1])

hots_df = pd.DataFrame (hots, columns = ['episode_name'])
hots_df['cold_open'] = np.zeros(12).astype(int)
hots_df = hots_df.replace(to_replace = 'Weight Loss', value = 'Weight Loss (Parts 1&2)')
data_ = pd.merge(data, hots_df, on='episode_name', how='left')
data_['cold_open'] = data_['cold_open'].fillna(1)

data_.loc[data_['cold_open'] == 1]

data = data_
data_explore = data.copy()
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_1348\1091642909.py in <module>
----> 1 with open('MLP_cold_opens.txt', 'r') as file:
      2     new_data = file.read().replace('\n', '')
      3 
      4 x = new_data.split(")")
      5 hots = []

FileNotFoundError: [Errno 2] No such file or directory: 'MLP_cold_opens.txt'
```

After making sure that all the necessary libraries or submodules are uploaded here, please follow the given skeleton to create your project report.

- Your completed assignment must follow this structure
- You should not add or remove any of these sections, if you feel it is necessary you may add extra subsections within each (such as *2.1. Encoding*).

Do not forget to remove the instructions for each section in the final document.

1. Introduction

This section should include a brief introduction to the task and the data (assume this is a report you are delivering to a client).

- If you use any additional data sources, you should introduce them here and discuss why they were included.
- Briefly outline the approaches being used and the conclusions that you are able to draw.

2. Exploratory Data Analysis and Feature Engineering

Include a detailed discussion of the data with a particular emphasis on the features of the data that are relevant for the subsequent modeling.

- Including visualizations of the data is strongly encouraged - all code and plots must also be described in the write up.
- Think carefully about whether each plot needs to be included in your final draft - your report should include figures but they should be as focused and impactful as possible.

Additionally, this section should also implement and describe any preprocessing / feature engineering of the data.

- Specifically, this should be any code that you use to generate new columns in the data frame `d`. All of this processing is explicitly meant to occur before we split the data in to training and testing subsets.
- Processing that will be performed as part of an sklearn pipeline can be mentioned here but should be implemented in the following section.*

All code and figures should be accompanied by text that provides an overview / context to what is being done or presented.

In [ ]:
```
print ("Columns in the df are:")
print (data_explore.columns.tolist(), "\n")

dup_num = pd.DataFrame(data_explore).duplicated().sum()
print("There are {} duplicated values in the data".format(dup_num))

na_num = data_explore.isna().sum().sum()
print(f"There are {na_num} Na values in the data")

data_explore.describe()
```

Measuring the correlation between the current numerical variables:

In [ ]:
```
sns.set(rc={'figure.figsize': (10, 6)})
sns.heatmap(data_explore.corr(), annot = True, fmt = '.2f', linewidths = 2)
plt.title("Correlation Heatmap for current numerical variables")
plt.show()
```

We will first drop the episode name and episode number features because they are not relevant to the model.

In [ ]:
```
data_explore = data_explore.drop("episode_name", axis=1)
data_explore = data_explore.drop("episode", axis=1)

print (data_explore.columns.tolist())
```

Next, we will use the variable "air_date" to create two new features: Categorical variable "calendar_season", being the season (autumn, winter, spring, summer) when the episode was aired; and (numerical variable?) "air_month" being the month the episode aired. After this we drop "air_date".

In [ ]:
```
# data['air_date'] = pd.to_datetime(data['air_date'])

# convert the 'dates' column to datetime format
data_explore['air_date'] = pd.to_datetime(data_explore['air_date'])

# create a new categorical column 'calendar_season' by assigning a category "winter",
# "spring", "summer", "autumn" depending on the air-month.
data_explore['calendar_season'] = pd.cut(
    data_explore['air_date'].dt.month,
    [0, 3, 6, 9, 12],
    labels=['winter', 'spring', 'summer', 'autumn']
)

# convert calender season to string column rather than categorical column.
data_explore['calendar_season'] = data_explore['calendar_season'].astype(str)

# Create month column.
data_explore['month'] = data_explore['air_date'].dt.month

data_explore.head(5)
```

In [ ]:
```
data_explore = data_explore.drop("air_date", axis=1)
data_explore.head(5)
```

We now write a one hot encoder function to assign a binary attribute per category for the Categorical variables ("director, "writer", "calendar_season" and "main_chars"):

In [ ]:
```
def OneHot_encoder(df, cat_variable):
    oneHot_df = df.copy()
    variable_initials = ''.join([word[0] for word in cat_variable.split("_")])


    unique_chars = set(';'.join(oneHot_df[cat_variable]).split(';'))

    # Create binary attributes for each unique character
    for char in unique_chars:
        oneHot_df[variable_initials+"_"+char] = oneHot_df[cat_variable].apply(lambda x: 1 if char in x else 0)


    # Drop the original main_chars column
    oneHot_df = oneHot_df.drop(columns=[cat_variable])

    return oneHot_df


cat_variables = ['director', 'writer', 'main_chars', 'calendar_season']

for cat_variable in cat_variables:
    data_explore = OneHot_encoder(data_explore, cat_variable)


data_explore.head(5)
```

By inspection we realised that there are some repeated directors where in some data entries the name of the director is spelled wrong. The specific cases were:

- Charles McDougall (correct name) and Charles McDougal
- Claire Scanlon (correct name) and Claire Scanlong
- Greg Daniels (correct name) and Greg Daneils
- Ken Whittingham (correct name) and Ken Wittingham
- Paul Lieberstein (correct name) and Paul Lieerstein

Thus we now write code to remove columns with the wrong name, where we will put a 1 in the correct name column if the incorrect one had a 1 and the correct one didn't.

In [37]:
```
correct_names = ['d_Charles McDougall','d_Claire Scanlon','d_Greg Daniels', 'd_Ken Whittingham', 'd_Paul Lieberstein']
wrong_names = ['d_Charles McDougal','d_Claire Scanlong','d_Greg Daneils', 'd_Ken Wittingham', 'd_Paul Lieerstein']

for i, correct_name in enumerate(correct_names):
    data_explore.loc[data_explore[wrong_names[i]] == 1, correct_name] = 1
    data_explore = data_explore.drop(wrong_names[i], axis=1)

data_explore.head(5)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tolerance)
   3628             try:
-> 3629                 return self._engine.get_loc(casted_key)
   3630             except KeyError as err:

~\Anaconda3\lib\site-packages\pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

~\Anaconda3\lib\site-packages\pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'd_Charles McDougal'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_1348\104851003.py in <module>
      3
      4 for i, correct_name in enumerate(correct_names):
----> 5     data_explore.loc[data_explore[wrong_names[i]] == 1, correct_name] = 1
      6     data_explore = data_explore.drop(wrong_names[i], axis=1)
      7

~\Anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self, key)
   3503             if self.columns.nlevels > 1:
   3504                 return self._getitem_multilevel(key)
-> 3505             indexer = self.columns.get_loc(key)
   3506             if is_integer(indexer):
   3507                 indexer = [indexer]

~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tolerance)
   3629                 return self._engine.get_loc(casted_key)
   3630             except KeyError as err:
-> 3631                 raise KeyError(key) from err
   3632             except TypeError:
   3633                 # If we have a listlike key, _check_indexing_error will raise

KeyError: 'd_Charles McDougal'
```

Adding some further data to the model

In the following cells we add some data to the model from Nasir Khalid's dataset (Khalid, 2021).

In [43]:
```
new_data = pd.read_csv("The-Office-Lines-V4.csv").copy()
new_data.head()
```

Out[43]:

	season	episode	title	scene	speaker	line	Unnamed: 6
0	1	1	Pilot	1	Michael	All right Jim. Your quarterlies look very good...	NaN
1	1	1	Pilot	1	Jim	Oh, I told you. I couldn't close it. So...	NaN
2	1	1	Pilot	1	Michael	So you've come to the master for guidance? Is ...	NaN
3	1	1	Pilot	1	Jim	Actually, you called me in here, but yeah.	NaN
4	1	1	Pilot	1	Michael	All right. Well, let me show you how it's done.	NaN

This dataset, as seen above, contains each line in the show, demarcated by speaker and scene. In the cell bellow, we aggregate the lines per character per episode, to give the total lines spoken by each character in each episode.

In [97]:
```
#Get a list of unique speakers from the speaker column
unique_speakers = new_data.speaker.unique()

#Create a dummy column which tracks episode, since season:episode are two columns
new_data["episode_code"] = new_data.season.astype(str)+"."+new_data.episode.astype(str)

#Add a dummy 1 to each line, so that the sum will count totals per character for each line
new_data["dummy"] = 1

new_data.head()

#By episode and speaker, sum the total number of lines
lines = new_data.groupby(["episode_code","speaker"]).sum()

#Need to now take the dummy sums and probably take the max (most-speaker) in an episode and maybe second most
#e.g. episode 1.1 has michael 81 lines and pam 40 lines so take 1.1 to have "most speech"="michael" and "second most"="pam"
lines.head(20)
```

Out[97]:

episode_code	speaker	season	episode	scene	dummy
1.1	Angela	1	1	14	1
	Dwight	29	29	693	29
	Jan	12	12	120	12
	Jim	36	36	982	36
	Kevin	1	1	14	1
	Man	1	1	26	1
	Michael	81	81	1594	81
	Michel	1	1	11	1
	Oscar	3	3	63	3
	Pam	40	40	1099	40
	Phyllis	2	2	38	2
	Roy	5	5	171	5
	Ryan	8	8	211	8
	Stanley	5	5	116	5
	Todd Packer	3	3	33	3
1.2	Angela	4	8	252	4
	Dwight	17	34	860	17
	Jim	25	50	1356	25
	Kelly	2	4	121	2
	Kevin	8	16	473	8

3. Model Fitting and Tuning

In this section you should detail your choice of model and describe the process used to refine and fit that model.

- You are strongly encouraged to explore many different modeling methods (e.g. linear regression, regression trees, lasso, etc.) but you should not include a detailed narrative of all of these attempts.
- At most this section should mention the methods explored and why they were rejected - most of your effort should go into describing the model you are using and your process for tuning and validatin it.

For example if you considered a linear regression model, a classification tree, and a lasso model and ultimately settled on the linear regression approach then you should mention that other two approaches were tried but do not include any of the code or any in depth discussion of these models beyond why they were rejected. This section should then detail is the development of the linear regression model in terms of features used, interactions considered, and any additional tuning and validation which ultimately led to your final model.

This section should also include the full implementation of your final model, including all necessary validation. As with figures, any included code must also be addressed in the text of the document.

4. Discussion and Conclusions

*In this section you should provide a general overview of **your final model**, its **performance**, and **reliability**.*

- You should discuss what the implications of your model are in terms of the included features, predictive performance, and anything else you think is relevant.

- This should be written with a target audience of a NBC Universal executive who is with the show and university level mathematics but not necessarily someone who has taken a postgraduate statistical modeling course.

- Your goal should be to convince this audience that your model is both accurate and useful.

- Finally, you should include concrete recommendations on what NBC Universal should do to make their reunion episode a popular as possible.

Keep in mind that a negative result, i.e. a model that does not work well predictively, but that is well explained and justified in terms of why it failed will likely receive higher marks than a model with strong predictive performance but with poor or incorrect explanations / justifications.

5. References

In this section, you should present a list of external sources (except the course materials) that you used during the project, if any

Additional data sources can be cited here, in addition to related python documentations, any other webpage sources that you benefited from

- Khalid, Nasir. 2021. *The Office (US) - Complete Dialogue/Transcript*, Kaggle. Accessed on 04/03/2023. URL: https://www.kaggle.com/datasets/nehaprabhavalkar/the-office-dataset