- Rescale the real valued features using any strategy you choose (StandardScaler, MinMaxScaler, Normalizer, etc)
- Augment at least one feature
- Implement a train-test split with 20% of the data going to the test data. Make sure that the test and train data are balanced in terms of the desired class.

After writing your preprocessing code, write out a description of what you did for each step and provide a justification for your choices. All descriptions should be written in the markdown cells of the jupyter notebook. Make sure your writing is clear and professional.

We highly recommend reading through the scikit-learn documentation to make this part easier.

[ ]:
```
data = pd.read_csv('datasets/hotel_booking.csv')
```

[ ]:
```
data.head(5)
```

[ ]:
```
   is_canceled         hotel  lead_time arrival_date_month  \
0            0  Resort Hotel          4           February
1            1    City Hotel        172               June
2            0    City Hotel          4           November
3            1    City Hotel         68          September
4            1    City Hotel        149               July

   stays_in_weekend_nights  stays_in_week_nights  adults  children  babies  \
0                        1                     2       2       0.0       0
1                        0                     2       1       0.0       0
2                        2                     1       1       0.0       0
3                        0                     2       2       0.0       0
4                        2                     4       3       0.0       0

  meal  … booking_changes  deposit_type  days_in_waiting_list  \
0   FB  …               0    No Deposit                     0
1   BB  …               0    No Deposit                     0
2   BB  …               0    No Deposit                     0
3   HB  …               0    No Deposit                     0
4   BB  …               0    No Deposit                     0

     customer_type    adr required_car_parking_spaces  \
0        Transient   75.0                           0
1  Transient-Party   95.0                           0
2        Transient   65.0                           0
3  Transient-Party    0.0                           0
4        Transient  167.7                           0

   total_of_special_requests             name                          email  \
0                          1      Linda Moore                 LMoore@att.com
1                          0   Madison Greene  Greene_Madison56@verizon.com
2                          0  Alicia Richards   Richards.Alicia@comcast.net
```

```
3                          0    Gregory Smith      GregorySmith@outlook.com
4                          0  Rachel Martinez          Rachel.M@outlook.com

   phone-number
0  217-602-3707
1  791-162-2669
2  442-385-2754
3  670-687-2703
4  692-194-2274

[5 rows x 24 columns]
```

[ ]:
```
data.isnull().sum()
```

[ ]:
```
is_canceled                       0
hotel                             0
lead_time                         0
arrival_date_month                0
stays_in_weekend_nights           0
stays_in_week_nights              0
adults                            0
children                          3
babies                            0
meal                              0
country                           0
previous_cancellations            0
previous_bookings_not_canceled    0
reserved_room_type                0
booking_changes                   0
deposit_type                      0
days_in_waiting_list              0
customer_type                     0
adr                               0
required_car_parking_spaces       0
total_of_special_requests         0
name                              0
email                             0
phone-number                      0
dtype: int64
```

[ ]:
```
data = data.dropna()
```

[ ]:
```
data.isnull().sum()
```

[ ]:
```
is_canceled                       0
hotel                             0
lead_time                         0
```

```
arrival_date_month                0
stays_in_weekend_nights           0
stays_in_week_nights              0
adults                            0
children                          0
babies                            0
meal                              0
country                           0
previous_cancellations            0
previous_bookings_not_canceled    0
reserved_room_type                0
booking_changes                   0
deposit_type                      0
days_in_waiting_list              0
customer_type                     0
adr                               0
required_car_parking_spaces       0
total_of_special_requests         0
name                              0
email                             0
phone-number                      0
dtype: int64
```

I checked if there were any null features (missing values). There were missing values for the children row. So I used imputation to replace the missing values with the median values. I then checked to make sure there were no more null values. Thus I didn't drop any fields.

[ ]:
```
data.describe()
```

[ ]:
```
        is_canceled     lead_time  stays_in_weekend_nights  \
count  78287.000000  78287.000000             78287.000000
mean       0.405789    109.264271                 0.882177
std        0.491047    113.690417                 0.986970
min        0.000000      0.000000                 0.000000
25%        0.000000     17.000000                 0.000000
50%        0.000000     71.000000                 1.000000
75%        1.000000    169.000000                 2.000000
max        1.000000    737.000000                16.000000

       stays_in_week_nights        adults      children        babies  \
count          78287.000000  78287.000000  78287.000000  78287.000000
mean               2.437953      1.838939      0.089312      0.008673
std                1.864411      0.615959      0.369761      0.104879
min                0.000000      0.000000      0.000000      0.000000
25%                1.000000      2.000000      0.000000      0.000000
50%                2.000000      2.000000      0.000000      0.000000
75%                3.000000      2.000000      0.000000      0.000000
```

```
max           41.000000     55.000000     10.000000     10.000000

       previous_cancellations  previous_bookings_not_canceled  \
count            78287.000000                    78287.000000
mean                 0.108703                        0.174652
std                  0.885154                        1.738106
min                  0.000000                        0.000000
25%                  0.000000                        0.000000
50%                  0.000000                        0.000000
75%                  0.000000                        0.000000
max                 26.000000                       72.000000

       booking_changes  days_in_waiting_list           adr  \
count     78287.000000          78287.000000  78287.000000
mean          0.202294              2.863260     98.157831
std           0.596073             19.670762     51.825338
min           0.000000              0.000000     -6.380000
25%           0.000000              0.000000     65.000000
50%           0.000000              0.000000     90.000000
75%           0.000000              0.000000    120.000000
max          20.000000            391.000000   5400.000000

       required_car_parking_spaces  total_of_special_requests
count                 78287.000000               78287.000000
mean                      0.065107                   0.509970
std                       0.248058                   0.768108
min                       0.000000                   0.000000
25%                       0.000000                   0.000000
50%                       0.000000                   0.000000
75%                       0.000000                   1.000000
max                       3.000000                   5.000000
```

[ ]:
```
categorical = ['hotel','arrival_date_month','meal', 'country',␣
 ↪'reserved_room_type','deposit_type','customer_type']
numerical =␣
 ↪['stays_in_weekend_nights','stays_in_week_nights','adults','children','babies','previous_ca
            'adr','required_car_parking_spaces','total_of_special_requests']
```

[ ]:
```
data = data.drop(["name","email","phone-number"], axis=1) # remove the␣
 ↪categorical features
```

I split the data into categorical and numerical. I dropped the name, email, and phone number fields because they are not relevant to whether or not a room is likely to be canceled.

[ ]:
```
from sklearn.base import BaseEstimator, TransformerMixin
```

```
# col indices

stays_in_weekend_nights = 2
stays_in_week_nights = 3
adults = 4
children = 5
babies = 6

class AugmentDataTransformer(BaseEstimator, TransformerMixin):
    def __init__(self):
        pass

    def fit(self, data, y=None):
        return self

    def transform(self, data):
        # Calculate Max_yearly_bookings
        # weekend_to_weekday_ratio = int(data[:,stays_in_weekend_nights] /␣
 ↪data[:,stays_in_week_nights])
        total_people = data[:,adults] + data[:,children] + data[:,babies]



        return np.c_[data, total_people]
```

```
[ ]: from sklearn.impute import SimpleImputer
     from sklearn.pipeline import Pipeline
     from sklearn.preprocessing import StandardScaler
     from sklearn.preprocessing import OneHotEncoder

     #pipeline for real valued features
     num_pipeline = Pipeline([
             ('imputer', SimpleImputer(strategy="median")), #Imputes using median
             ('std_scaler', StandardScaler()),
         ])


     #Full Pipeline


     #Applies different transformations on numerical columns vs categorial columns
     full_pipeline = ColumnTransformer([
             ("num", num_pipeline, numerical),
             ("cat", OneHotEncoder(), categorical),
         ])
```

I have a feature that measures the total number of people staying in the room, including adults, children, and babies. I think the number of people may be indicative of whether or not the room will be canceled or not. For the weekend to weekday ratio, this also may help indicate whether the room will be canceled or not.

I used one hot encoding for categorical features. I used standard scaler for rescaling the real valued features.

```
[ ]: y = data["is_canceled"]
     x = data.drop(["is_canceled"],axis = 1)
```

```
[ ]: train, test, target, target_test = train_test_split(x,y, test_size=0.2,␣
      ↪stratify= y, random_state=0)
```

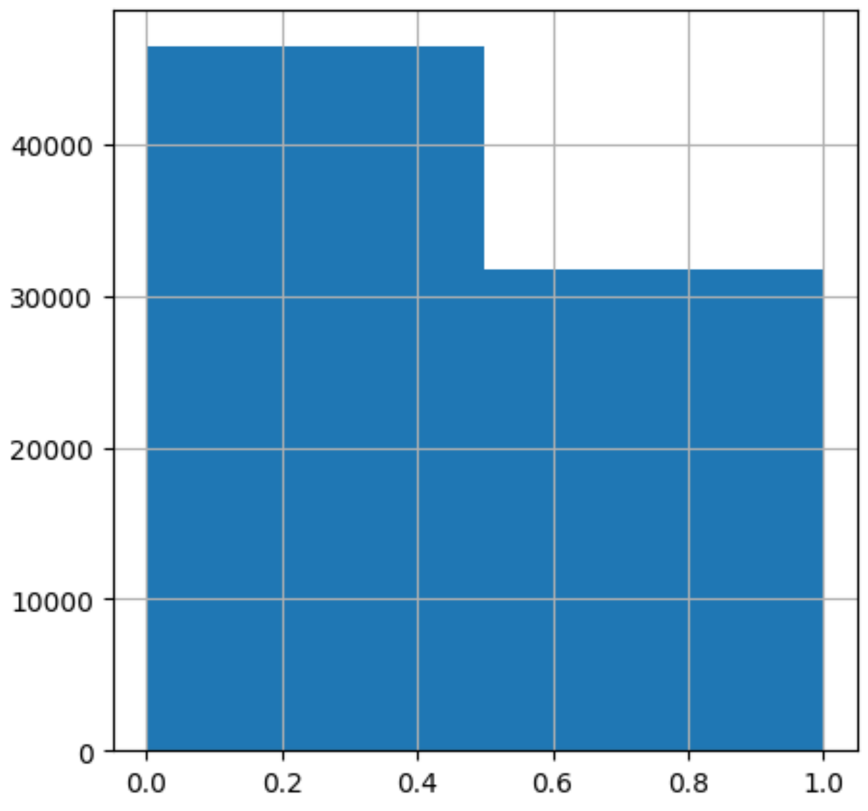
```
[ ]: print(list(train))
```

```
['hotel', 'lead_time', 'arrival_date_month', 'stays_in_weekend_nights',
'stays_in_week_nights', 'adults', 'children', 'babies', 'meal', 'country',
'previous_cancellations', 'previous_bookings_not_canceled',
'reserved_room_type', 'booking_changes', 'deposit_type', 'days_in_waiting_list',
'customer_type', 'adr', 'required_car_parking_spaces',
'total_of_special_requests']
```

```
[ ]: train = full_pipeline.fit_transform(train)
     test = full_pipeline.transform(test)
```

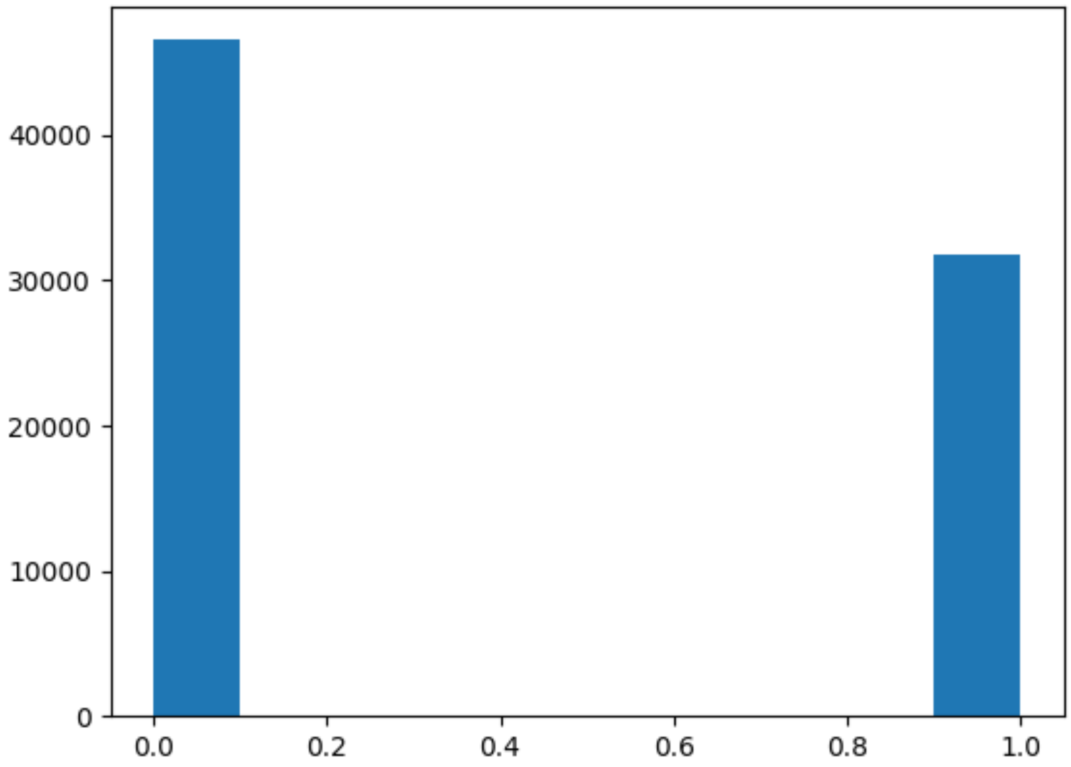
```
[ ]: data['is_canceled'].hist(bins=2, figsize=(5,5))
     data['is_canceled'].value_counts()
```

```
[ ]: 0    46519
     1    31768
     Name: is_canceled, dtype: int64
```



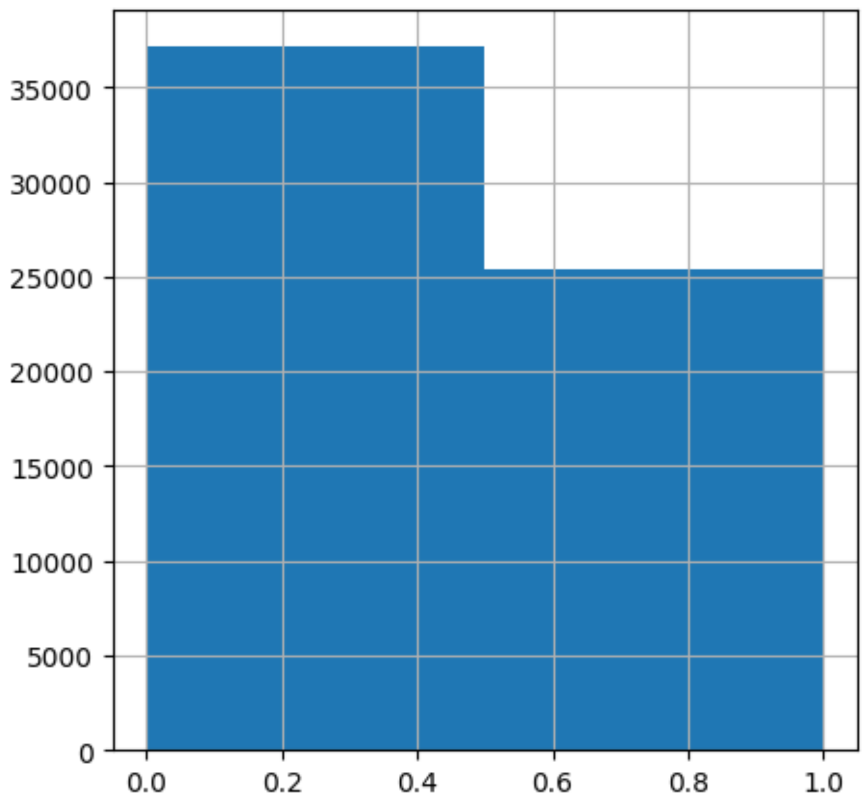
```
[ ]: plt.hist(data['is_canceled'])
```

```
[ ]: (array([46519.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
                 0., 31768.]),
      array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
      <BarContainer object of 10 artists>)
```



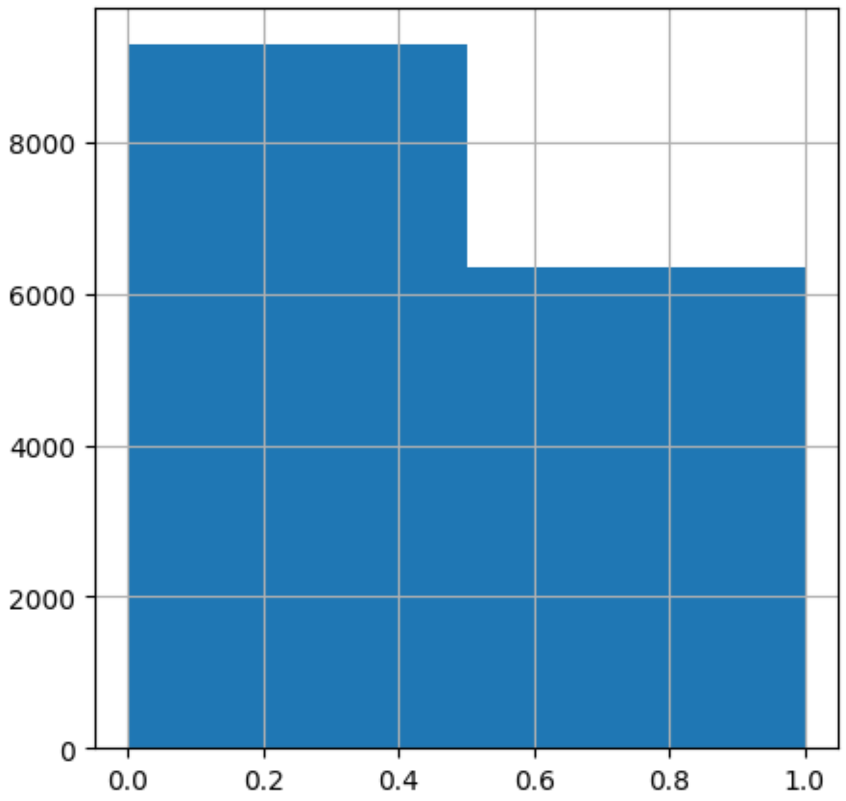
```
[ ]: #Training classes
     target.hist(bins=2, figsize=(5,5))
     target.value_counts()
```

```
[ ]: 0    37215
     1    25414
     Name: is_canceled, dtype: int64
```



```
[ ]: #Testing classes
     target_test.hist(bins=2, figsize=(5,5))
     target_test.value_counts()
```

```
[ ]: 0    9304
     1    6354
     Name: is_canceled, dtype: int64
```



I set the target feature to be is_canceled. I implemented a train-test split with 20% of data going to test. The imbalance is not heavy, it is similar to the data from project 2, so there is no need to performance balancing techniques on the dataset. After the split, the balance remains the same which is good. We can use this as a baseline: 6354 / (9304 + 6354) = 0.405798952612, which measures the probability that the room will be canceled.

In summary,

- I used one-hot encoding for cateogrical features. For the features with multiple values, I didn't do anything special.
- I dropped the name, email, and phone number fields since I do not believe they correlate with whether or not a room is canceled.
- I handled the missing children values with median imputation.
- I rescaled the real valued features with the standard scalar since I believe this is a standard and popular choice.
- I augmented one feature, a total number of people.
- I made the train-test split and ensured that the target data is reasonably balanced.

2.2 (50 pts) Try out a few models

Now that you have pre-processed your data, you are ready to try out different models.

For this part of the project, we want you to experiment with all the different models demonstrated in the course to determine which one performs best on the dataset.

You must perform classification using at least 3 of the following models: - Logistic Regression - K-nearest neighbors - SVM - Decision Tree - Multi-Layer Perceptron

Due to the size of the dataset, be careful which models you use and look at their documentation to see how you should tackle this size issue for each model.

For full credit, you must perform some hyperparameter optimization on your models of choice. You may find the following scikit-learn library on hyperparameter optimization useful.

For each model chosen, write a description of which models were chosen, which parameters you optimized, and which parameters you choose for your best model. While the previous part of the project asked you to pre-process the data in a specific manner, you may alter pre-processing step as you wish to adjust for your chosen classification models.

[ ]:
```
import joblib
```

[ ]:
```
%ls
```

```
'Copy of Project_3.ipynb'   datasets/         mlp.pkl
 dataset.csv                jupyter_images/   Project_3.ipynb
```

[ ]:
```
loaded_model = joblib.load('mlp.pkl')
```

2.3 Extra Credit

We have provided an extra test dataset named "hotel_booking_test.csv" that does not have the target labels. Classify the samples in the dataset with your best model and write them into a csv file. Submit your csv file to our Kaggle contest. The website will specify your classification accuracy on the test set. We will award a bonus point for the project for every percentage point over 75% that you get on your kaggle test accuracy.

To get the bonus points, you must also write out a summary of the model that you submit including any changes you made to the pre-processing steps. The summary must be written in a markdown cell of the jupyter notebook. Note that you should not change earlier parts of the project to complete the extra credit.

Kaggle Submission Instruction Submit a two column csv where the first column is named "ID" and is the row number. The second column is named "target" and is the classification for each sample. Make sure that the sample order is preserved.

Summary of model: clf = MLPClassifier(hidden_layer_sizes=(100,100), max_iter = 800, random_state=0). I used an MLP model with hidden layer sizes (100,100), 800 max iterations, and the rest of the parameters set at default. I achieved 85% test accuracy. I did not change any of the pre-processing steps. I also tried using a decision tree model with and without PCA and both did not achieve as high an accuracy compared to the MLP model. I didn't change any previous parts of the work.

```
[ ]: data2 = pd.read_csv('datasets/hotel_booking_test.csv')
```

```
[ ]: print(data2.shape)
```

```
(8699, 23)
```

```
[ ]: data2.head()
```

```
[ ]:          hotel  lead_time arrival_date_month  stays_in_weekend_nights  \
     0    City Hotel        177             August                        0
     1  Resort Hotel        217             August                        2
     2    City Hotel         65          September                        2
     3    City Hotel        377            October                        0
     4    City Hotel         75                May                        2

        stays_in_week_nights  adults  children  babies meal country  …  \
     0                     2       2       0.0       0   SC     FRA  …
     1                     5       2       1.0       0   HB     PRT  …
     2                     1       2       0.0       0   HB     PRT  …
     3                     2       2       0.0       0   HB     DEU  …
     4                     1       2       0.0       0   BB     PRT  …

        booking_changes  deposit_type days_in_waiting_list    customer_type    adr  \
     0                0    No Deposit                    0        Transient   94.5
     1                0    No Deposit                    0        Transient  170.0
     2                0    Non Refund                    0        Transient   86.0
     3                0    No Deposit                    0  Transient-Party  115.0
     4                0    No Deposit                    0  Transient-Party   90.0

        required_car_parking_spaces total_of_special_requests            name  \
     0                           0                         1  Wendy Moore DDS
     1                           0                         0   Brandon Fields
     2                           0                         0      Chad Peters
     3                           0                         1     Colin Rosario
     4                           0                         0       Dana Booth

                             email  phone-number
     0             Wendy_D50@att.com  350-596-1114
     1              BFields@zoho.com  872-706-9025
     2        Peters.Chad@outlook.com  581-621-8941
     3  Rosario_Colin@protonmail.com  783-552-8468
     4       Dana_Booth22@outlook.com  194-105-9813

     [5 rows x 23 columns]
```

```
[ ]: #Drop columns
     data2 = data2.drop(["name", "email","phone-number"],axis= 1)
```

```
[ ]: print(list(data2))
```

```
['hotel', 'lead_time', 'arrival_date_month', 'stays_in_weekend_nights',
'stays_in_week_nights', 'adults', 'children', 'babies', 'meal', 'country',
'previous_cancellations', 'previous_bookings_not_canceled',
'reserved_room_type', 'booking_changes', 'deposit_type', 'days_in_waiting_list',
'customer_type', 'adr', 'required_car_parking_spaces',
'total_of_special_requests']
```

```
[ ]: categorical_features = data2.select_dtypes(include=['object']).columns.tolist()
     numerical_features = data2.select_dtypes(include=['int64', 'float64']).columns.
      ↪tolist()
     print(categorical_features)
     print(numerical_features)
```

```
['hotel', 'arrival_date_month', 'meal', 'country', 'reserved_room_type',
'deposit_type', 'customer_type']
['lead_time', 'stays_in_weekend_nights', 'stays_in_week_nights', 'adults',
'children', 'babies', 'previous_cancellations',
'previous_bookings_not_canceled', 'booking_changes', 'days_in_waiting_list',
'adr', 'required_car_parking_spaces', 'total_of_special_requests']
```

```
[ ]: print(len(data2))
```

```
8699
```

```
[ ]: data3 = data2.dropna()
```

```
[ ]: print(len(data3))
```

```
8698
```

```
[ ]: median = data2["children"].median()
     data2["children"].fillna(median, inplace=True) # option 3: replace na values␣
      ↪with median values
```

```
[ ]: data2.isnull().sum()
```

```
[ ]: hotel                          0
     lead_time                      0
     arrival_date_month             0
     stays_in_weekend_nights        0
     stays_in_week_nights           0
     adults                         0
     children                       0
```

```
babies                            0
meal                              0
country                           0
previous_cancellations            0
previous_bookings_not_canceled    0
reserved_room_type                0
booking_changes                   0
deposit_type                      0
days_in_waiting_list              0
customer_type                     0
adr                               0
required_car_parking_spaces       0
total_of_special_requests         0
dtype: int64
```

[ ]:
```python
categorical = data2.select_dtypes(include='object')
numerical = data2.select_dtypes(include='number')
print(len(list(categorical)))
print(len(list(numerical)))
```

```
7
13
```

[ ]:
```python
 # One hot encoding features
# from sklearn.preprocessing import LabelEncoder

# le = LabelEncoder()
# for feature in categorical:
#   data2[feature] = le.fit_transform(data2[feature])
#   print(le.classes_)
```

[ ]:
```python
print(len(data2))
```

```
8699
```

[ ]:
```python
output = full_pipeline.transform(data2)
```

[ ]:
```python
loaded_model.fit(train, target)
```

[ ]:
```
MLPClassifier(hidden_layer_sizes=(100, 100), max_iter=800, random_state=0)
```

[ ]:
```python
predicted = loaded_model.predict(output)
```

[ ]:
```python
print(len(data2))
```

```
8699
```

[ ]:
```

```

[ ]:
```
print(list(data2))
```

```
['hotel', 'lead_time', 'arrival_date_month', 'stays_in_weekend_nights',
'stays_in_week_nights', 'adults', 'children', 'babies', 'meal', 'country',
'previous_cancellations', 'previous_bookings_not_canceled',
'reserved_room_type', 'booking_changes', 'deposit_type', 'days_in_waiting_list',
'customer_type', 'adr', 'required_car_parking_spaces',
'total_of_special_requests']
```

[ ]:
```
dataset = pd.DataFrame({'target': predicted})
```

[ ]:
```
dataset = dataset.rename_axis('ID')
```

[ ]:
```
dataset.head()
```

[ ]:
```
    target
ID
0        1
1        1
2        1
3        0
4        1
```

[ ]:
```
dataset['ID']=dataset.index
```

[ ]:
```
dataset = dataset.reset_index(drop=True)
```

[ ]:
```
dataset = dataset[['ID', 'target']]
```

[ ]:
```
dataset.head()
```

[ ]:
```
   ID  target
0   0       1
1   1       1
2   2       1
3   3       0
4   4       1
```

[ ]:
```
dataset.to_csv('dataset.csv', index=False)
```

[ ]:
```
print(len(dataset))
```

```
8699
```

[103]:
```
!apt-get install texlive-xetex texlive-fonts-recommended texlive-plain-generic
!jupyter nbconvert --to pdf ECProject_3.ipynb
```

```
Reading package lists… Done
Building dependency tree
Reading state information… Done
texlive-fonts-recommended is already the newest version (2019.20200218-1).
texlive-plain-generic is already the newest version (2019.202000218-1).
texlive-xetex is already the newest version (2019.20200218-1).
0 upgraded, 0 newly installed, 0 to remove and 24 not upgraded.
[NbConvertApp] WARNING | pattern 'ECProject_3.ipynb' matched no files
This application is used to convert notebook files (*.ipynb)
        to various other formats.

        WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

Options
=======
The options below are convenience aliases to configurable class-options,
as listed in the "Equivalent to" description-line of the aliases.
To see all configurable class-options for some <cmd>, use:
    <cmd> --help-all

--debug
    set log level to logging.DEBUG (maximize logging output)
    Equivalent to: [--Application.log_level=10]
--show-config
    Show the application's configuration (human-readable format)
    Equivalent to: [--Application.show_config=True]
--show-config-json
    Show the application's configuration (json format)
    Equivalent to: [--Application.show_config_json=True]
--generate-config
    generate default config file
    Equivalent to: [--JupyterApp.generate_config=True]
-y
    Answer yes to any questions instead of prompting.
    Equivalent to: [--JupyterApp.answer_yes=True]
--execute
    Execute the notebook prior to export.
    Equivalent to: [--ExecutePreprocessor.enabled=True]
--allow-errors
    Continue notebook execution even if one of the cells throws an error and
include the error message in the cell output (the default behaviour is to abort
conversion). This flag is only relevant if '--execute' was specified, too.
    Equivalent to: [--ExecutePreprocessor.allow_errors=True]
--stdin
```

```
    read a single notebook file from stdin. Write the resulting notebook with
default basename 'notebook.*'
    Equivalent to: [--NbConvertApp.from_stdin=True]
--stdout
    Write notebook output to stdout instead of files.
    Equivalent to: [--NbConvertApp.writer_class=StdoutWriter]
--inplace
    Run nbconvert in place, overwriting the existing notebook (only
            relevant when converting to notebook format)
    Equivalent to: [--NbConvertApp.use_output_suffix=False
--NbConvertApp.export_format=notebook --FilesWriter.build_directory=]
--clear-output
    Clear output of current file and save in place,
            overwriting the existing notebook.
    Equivalent to: [--NbConvertApp.use_output_suffix=False
--NbConvertApp.export_format=notebook --FilesWriter.build_directory=
--ClearOutputPreprocessor.enabled=True]
--no-prompt
    Exclude input and output prompts from converted document.
    Equivalent to: [--TemplateExporter.exclude_input_prompt=True
--TemplateExporter.exclude_output_prompt=True]
--no-input
    Exclude input cells and output prompts from converted document.
            This mode is ideal for generating code-free reports.
    Equivalent to: [--TemplateExporter.exclude_output_prompt=True
--TemplateExporter.exclude_input=True
--TemplateExporter.exclude_input_prompt=True]
--allow-chromium-download
    Whether to allow downloading chromium if no suitable version is found on the
system.
    Equivalent to: [--WebPDFExporter.allow_chromium_download=True]
--disable-chromium-sandbox
    Disable chromium security sandbox when converting to PDF..
    Equivalent to: [--WebPDFExporter.disable_sandbox=True]
--show-input
    Shows code input. This flag is only useful for dejavu users.
    Equivalent to: [--TemplateExporter.exclude_input=False]
--embed-images
    Embed the images as base64 dataurls in the output. This flag is only useful
for the HTML/WebPDF/Slides exports.
    Equivalent to: [--HTMLExporter.embed_images=True]
--sanitize-html
    Whether the HTML in Markdown cells and cell outputs should be sanitized..
    Equivalent to: [--HTMLExporter.sanitize_html=True]
--log-level=<Enum>
    Set the log level by value or name.
    Choices: any of [0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR',
'CRITICAL']
```

```
    Default: 30
    Equivalent to: [--Application.log_level]
--config=<Unicode>
    Full path of a config file.
    Default: ''
    Equivalent to: [--JupyterApp.config_file]
--to=<Unicode>
    The export format to be used, either one of the built-in formats
            ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook',
'pdf', 'python', 'rst', 'script', 'slides', 'webpdf']
            or a dotted object name that represents the import path for an
            ``Exporter`` class
    Default: ''
    Equivalent to: [--NbConvertApp.export_format]
--template=<Unicode>
    Name of the template to use
    Default: ''
    Equivalent to: [--TemplateExporter.template_name]
--template-file=<Unicode>
    Name of the template file to use
    Default: None
    Equivalent to: [--TemplateExporter.template_file]
--theme=<Unicode>
    Template specific theme(e.g. the name of a JupyterLab CSS theme distributed
    as prebuilt extension for the lab template)
    Default: 'light'
    Equivalent to: [--HTMLExporter.theme]
--sanitize_html=<Bool>
    Whether the HTML in Markdown cells and cell outputs should be sanitized.This
    should be set to True by nbviewer or similar tools.
    Default: False
    Equivalent to: [--HTMLExporter.sanitize_html]
--writer=<DottedObjectName>
    Writer class used to write the
                                    results of the conversion
    Default: 'FilesWriter'
    Equivalent to: [--NbConvertApp.writer_class]
--post=<DottedOrNone>
    PostProcessor class used to write the
                                    results of the conversion
    Default: ''
    Equivalent to: [--NbConvertApp.postprocessor_class]
--output=<Unicode>
    overwrite base name use for output files.
                can only be used when converting one notebook at a time.
    Default: ''
    Equivalent to: [--NbConvertApp.output_base]
--output-dir=<Unicode>
```

```
    Directory to write output(s) to. Defaults
                                  to output to the directory of each notebook.
To recover
                                  previous default behaviour (outputting to the
current
                                  working directory) use . as the flag value.
    Default: ''
    Equivalent to: [--FilesWriter.build_directory]
--reveal-prefix=<Unicode>
    The URL prefix for reveal.js (version 3.x).
            This defaults to the reveal CDN, but can be any url pointing to a
copy
            of reveal.js.
            For speaker notes to work, this must be a relative path to a local
            copy of reveal.js: e.g., "reveal.js".
            If a relative path is given, it must be a subdirectory of the
            current directory (from which the server is run).
            See the usage documentation
            (https://nbconvert.readthedocs.io/en/latest/usage.html#reveal-js-
html-slideshow)
            for more details.
    Default: ''
    Equivalent to: [--SlidesExporter.reveal_url_prefix]
--nbformat=<Enum>
    The nbformat version to write.
            Use this to downgrade notebooks.
    Choices: any of [1, 2, 3, 4]
    Default: 4
    Equivalent to: [--NotebookExporter.nbformat_version]

Examples
--------

    The simplest way to use nbconvert is

            > jupyter nbconvert mynotebook.ipynb --to html

            Options include ['asciidoc', 'custom', 'html', 'latex', 'markdown',
'notebook', 'pdf', 'python', 'rst', 'script', 'slides', 'webpdf'].

            > jupyter nbconvert --to latex mynotebook.ipynb

            Both HTML and LaTeX support multiple output templates. LaTeX
includes
            'base', 'article' and 'report'.  HTML includes 'basic', 'lab' and
            'classic'. You can specify the flavor of the format used.

            > jupyter nbconvert --to html --template lab mynotebook.ipynb
            You can also pipe the output to stdout, rather than a file

            > jupyter nbconvert mynotebook.ipynb --stdout

            PDF is generated via latex

            > jupyter nbconvert mynotebook.ipynb --to pdf

            You can get (and serve) a Reveal.js-powered slideshow

            > jupyter nbconvert myslides.ipynb --to slides --post serve

            Multiple notebooks can be given at the command line in a couple of
            different ways:

            > jupyter nbconvert notebook*.ipynb
            > jupyter nbconvert notebook1.ipynb notebook2.ipynb

            or you can specify the notebooks list in a config file, containing::

                c.NbConvertApp.notebooks = ["my_notebook.ipynb"]

            > jupyter nbconvert --config mycfg.py

To see all available configurables, use `--help-all`.
```