Assignment 6 - Final Model Deployment

July 19, 2021

0.1 Assignment 6: Final Model Deployment

After running several other models and evaluating their performance we found that our original decision tree model still produced the highest accuracy. Our baseline accuracy was calculated using the total amount of patients in each length of stay at a hospital, with the highest being the group staying for 21-30 days at 27.48%. Of the three models that we selected, two performed higher than our baseline accuracy with lowest performing model being the logistic regression model at 26.5% accuracy, while the the decision tree and random forest models performed at 40.82% and 37.43% respectively. Our best performing decision tree model parameters has a max depth of 10, using the gini index, and the random forest model had an optimal amount of trees or estimators at 100, which also used the gini index. After tuning the hyperparameters for the Logistic regression surrounding the penalty type, solvers and max iterations it did not yeild better results than the decision tree or random forest model accuracy.

Deployed App: https://st-udteam1.herokuapp.com/
Git Repo: https://github.com/sivakodali/healthcareml

[2]:
```
import os
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn import preprocessing
import matplotlib.pyplot as plt
import csv
```

[3]:
```
# train = np.genfromtxt('train.csv', delimiter=',', dtype=None)
# test = np.genfromtxt('test.csv', delimiter=',', dtype=None)
df = pd.read_csv("train.csv")
```

0.2 EDA

[4]:
```
print(df.shape)
```

```
(318438, 18)
```

[5]:
```
df.head(10)
```

[5]:
```
   case_id  Hospital_code Hospital_type_code  City_Code_Hospital  \
0        1              8                  c                   3
1        2              2                  c                   5
```

2	3	10	e	1
3	4	26	b	2
4	5	26	b	2
5	6	23	a	6
6	7	32	f	9
7	8	23	a	6
8	9	1	d	10
9	10	10	e	1

	Hospital_region_code	Available Extra Rooms in Hospital	Department \
0	Z	3	radiotherapy
1	Z	2	radiotherapy
2	X	2	anesthesia
3	Y	2	radiotherapy
4	Y	2	radiotherapy
5	X	2	anesthesia
6	Y	1	radiotherapy
7	X	4	radiotherapy
8	Y	2	gynecology
9	X	2	gynecology

	Ward_Type	Ward_Facility_Code	Bed Grade	patientid	City_Code_Patient \
0	R	F	2.0	31397	7.0
1	S	F	2.0	31397	7.0
2	S	E	2.0	31397	7.0
3	R	D	2.0	31397	7.0
4	S	D	2.0	31397	7.0
5	S	F	2.0	31397	7.0
6	S	B	3.0	31397	7.0
7	Q	F	3.0	31397	7.0
8	R	B	4.0	31397	7.0
9	S	E	3.0	31397	7.0

	Type of Admission	Severity of Illness	Visitors with Patient	Age \
0	Emergency	Extreme	2	51-60
1	Trauma	Extreme	2	51-60
2	Trauma	Extreme	2	51-60
3	Trauma	Extreme	2	51-60
4	Trauma	Extreme	2	51-60
5	Trauma	Extreme	2	51-60
6	Emergency	Extreme	2	51-60
7	Trauma	Extreme	2	51-60
8	Trauma	Extreme	2	51-60
9	Trauma	Extreme	2	51-60

	Admission_Deposit	Stay
0	4911.0	0-10

```
1             5954.0  41-50
2             4745.0  31-40
3             7272.0  41-50
4             5558.0  41-50
5             4449.0  11-20
6             6167.0   0-10
7             5571.0  41-50
8             7223.0  51-60
9             6056.0  31-40
```

[6]:
```
# basic shape, data type, null values
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 318438 entries, 0 to 318437
Data columns (total 18 columns):
 #   Column                             Non-Null Count   Dtype
---  ------                             --------------   -----
 0   case_id                            318438 non-null  int64
 1   Hospital_code                      318438 non-null  int64
 2   Hospital_type_code                 318438 non-null  object
 3   City_Code_Hospital                 318438 non-null  int64
 4   Hospital_region_code               318438 non-null  object
 5   Available Extra Rooms in Hospital  318438 non-null  int64
 6   Department                         318438 non-null  object
 7   Ward_Type                          318438 non-null  object
 8   Ward_Facility_Code                 318438 non-null  object
 9   Bed Grade                          318325 non-null  float64
 10  patientid                          318438 non-null  int64
 11  City_Code_Patient                  313906 non-null  float64
 12  Type of Admission                  318438 non-null  object
 13  Severity of Illness                318438 non-null  object
 14  Visitors with Patient              318438 non-null  int64
 15  Age                                318438 non-null  object
 16  Admission_Deposit                  318438 non-null  float64
 17  Stay                               318438 non-null  object
dtypes: float64(3), int64(6), object(9)
memory usage: 43.7+ MB
```

[7]:
```
le = preprocessing.LabelEncoder()
```

[8]:
```
department_encoded=le.fit_transform(df.Department)
age_encoded = le.fit_transform(df.Age)
admission_encoded=le.fit_transform(df['Type of Admission'])
stay_encoded = le.fit_transform(df.Stay)
severity_encoded = le.fit_transform(df['Severity of Illness'])
ward_type_encoded = le.fit_transform(df['Ward_Type'])
ward_facility_encoded = le.fit_transform(df['Ward_Facility_Code'])
```

```
hospital_region_encoded = le.fit_transform(df['Hospital_region_code'])
hospital_type_encoded = le.fit_transform(df['Hospital_type_code'])
```

[9]:
```
df['department_encoded'] = department_encoded
df['age_encoded'] = age_encoded
df['admission_encoded'] = admission_encoded
df['stay_encoded'] = stay_encoded
df['severity_encoded'] = severity_encoded
df['ward_type_encoded'] = ward_type_encoded
df['ward_facility_encoded'] = ward_facility_encoded
df['hospital_region_encoded'] = hospital_region_encoded
df['hospital_type_encoded'] = hospital_type_encoded
```

[10]:
```
df.columns
```

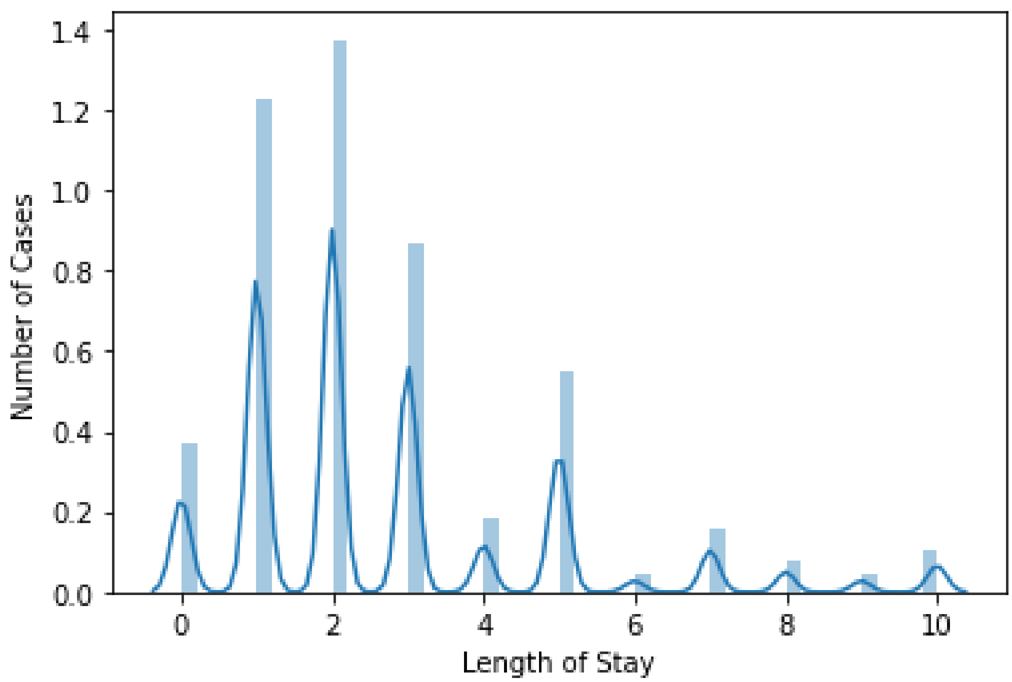
[10]:
```
Index(['case_id', 'Hospital_code', 'Hospital_type_code', 'City_Code_Hospital',
       'Hospital_region_code', 'Available Extra Rooms in Hospital',
       'Department', 'Ward_Type', 'Ward_Facility_Code', 'Bed Grade',
       'patientid', 'City_Code_Patient', 'Type of Admission',
       'Severity of Illness', 'Visitors with Patient', 'Age',
       'Admission_Deposit', 'Stay', 'department_encoded', 'age_encoded',
       'admission_encoded', 'stay_encoded', 'severity_encoded',
       'ward_type_encoded', 'ward_facility_encoded', 'hospital_region_encoded',
       'hospital_type_encoded'],
      dtype='object')
```

[11]:
```
# Checking for correlation amongst features
corrMatrix = df.corr()
corrMatrix.style.background_gradient()
```

[11]:
```
<pandas.io.formats.style.Styler at 0x21f6aed1f40>
```

[12]:
```
print("\nstay_encoded \n")
sns.distplot(stay_encoded)
plt.xlabel('Length of Stay')
plt.ylabel('Number of Cases')
plt.savefig('Length of Stay Value Counts')
plt.show()
print(df.Stay.unique())
print(df.stay_encoded.unique())
# df[['stay_encoded', 'stay']]
```

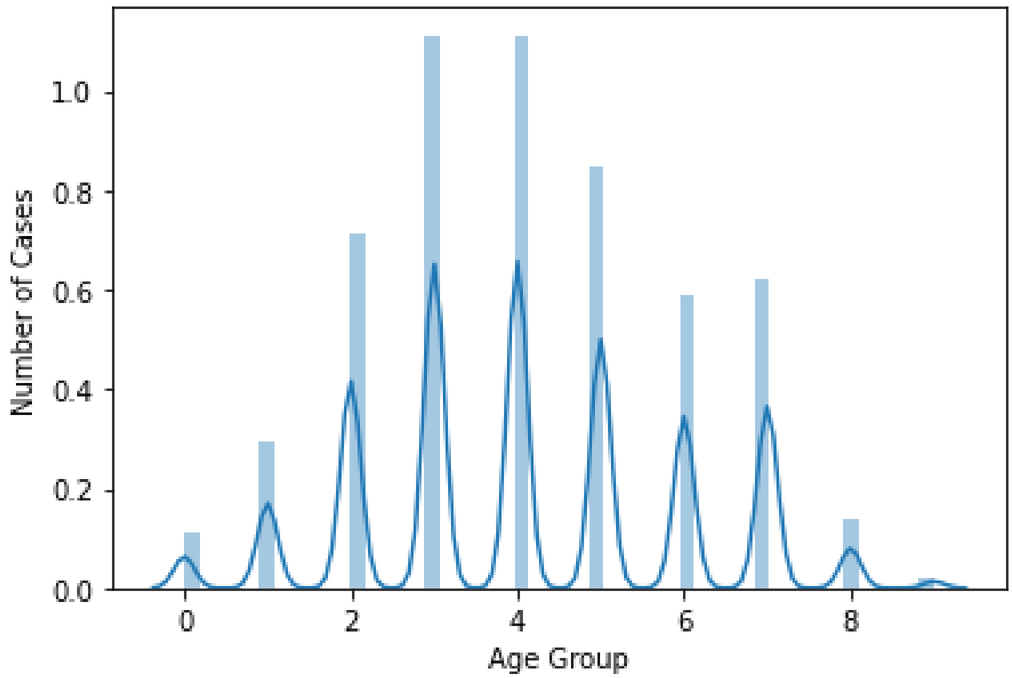
```
stay_encoded
```




```
['0-10' '41-50' '31-40' '11-20' '51-60' '21-30' '71-80'
 'More than 100 Days' '81-90' '61-70' '91-100']
[ 0  4  3  1  5  2  7 10  8  6  9]
```

[16]:
```python
print("age_encoded \n")
sns.distplot(age_encoded)
plt.xlabel('Age Group')
plt.ylabel('Number of Cases')
plt.savefig('Age Distribution')
plt.show()
print(df.Age.unique())
print(df.age_encoded.unique())
# df[['age_encoded', 'Age']]
```

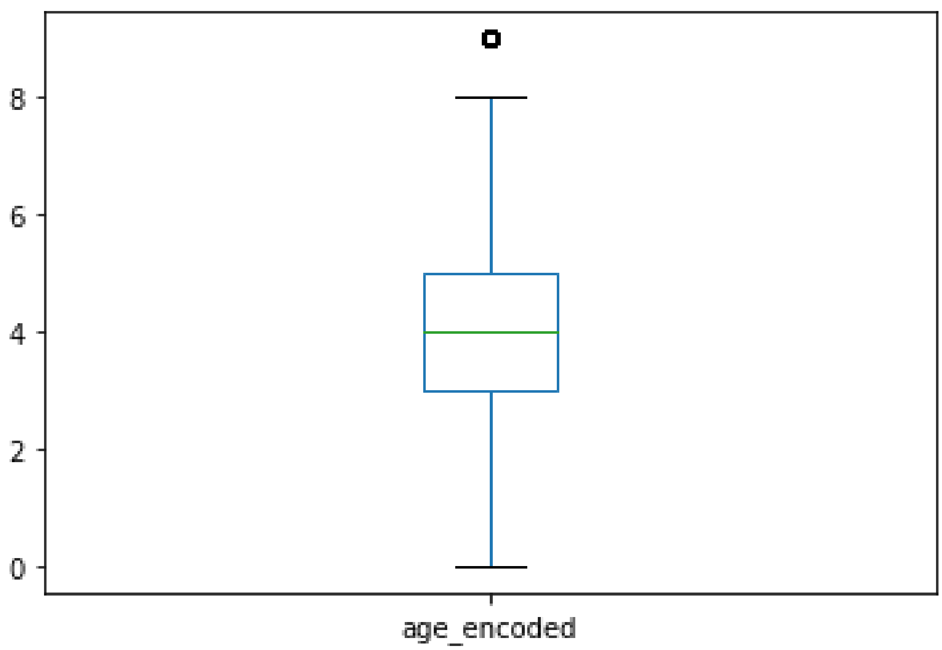
```
age_encoded
```




```
['51-60' '71-80' '31-40' '41-50' '81-90' '61-70' '21-30' '11-20' '0-10'
 '91-100']
[5 7 3 4 8 6 2 1 0 9]
```

[13]:
```
df['age_encoded'].plot.box()
```

[13]: <matplotlib.axes._subplots.AxesSubplot at 0x21f6bd5c040>



[18]:
```
# Prepare the data by separating X and y
# dropping unimportant features, such as passenger id, name, ticket number and␣
↪cabin number
# note that interesting features might be engieered from the dropped features␣
↪above

# axis = 1 below means dropping by columns, 0 means by rows
X = df.drop(['Stay', 'case_id','department_encoded', 'age_encoded',␣
↪'admission_encoded', 'stay_encoded', 'severity_encoded',␣
↪'ward_type_encoded', 'ward_facility_encoded',␣
↪'hospital_region_encoded','hospital_type_encoded'], axis=1)
y = df['Stay']
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 318438 entries, 0 to 318437
Data columns (total 16 columns):
 #   Column                             Non-Null Count   Dtype
---  ------                             --------------   -----
 0   Hospital_code                      318438 non-null  int64
 1   Hospital_type_code                 318438 non-null  object
 2   City_Code_Hospital                 318438 non-null  int64
 3   Hospital_region_code               318438 non-null  object
 4   Available Extra Rooms in Hospital  318438 non-null  int64
 5   Department                         318438 non-null  object
 6   Ward_Type                          318438 non-null  object
```

```
 7   Ward_Facility_Code                 318438 non-null  object
 8   Bed Grade                          318325 non-null  float64
 9   patientid                          318438 non-null  int64
 10  City_Code_Patient                  313906 non-null  float64
 11  Type of Admission                  318438 non-null  object
 12  Severity of Illness                318438 non-null  object
 13  Visitors with Patient              318438 non-null  int64
 14  Age                                318438 non-null  object
 15  Admission_Deposit                  318438 non-null  float64
dtypes: float64(3), int64(5), object(8)
memory usage: 38.9+ MB
```

[8]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[9]:
```
# any missing values?
X_train.isnull().sum()
```

[9]:
```
Hospital_code                         0
Hospital_type_code                    0
City_Code_Hospital                    0
Hospital_region_code                  0
Available Extra Rooms in Hospital     0
Department                            0
Ward_Type                             0
Ward_Facility_Code                    0
Bed Grade                            94
patientid                             0
City_Code_Patient                  3654
Type of Admission                     0
Severity of Illness                   0
Visitors with Patient                 0
Age                                   0
Admission_Deposit                     0
dtype: int64
```

0.3 Data Preprocessing

[10]:
```
#Original
num_features = ['Available Extra Rooms in Hospital', 'Admission_Deposit',␣
↪'Visitors with Patient']
cat_features = ['Hospital_code', 'City_Code_Hospital', 'Department',␣
↪'Ward_Type', 'City_Code_Patient', 'Type of Admission', 'Severity of␣
↪Illness', 'Age', 'Bed Grade', 'Hospital_type_code', 'Ward_Facility_Code']
```

[11]:

```python
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder

# Create the preprocessing pipeline for numerical features
# There are two steps in this pipeline
# Pipeline(steps=[(name1, transform1), (name2, transform2), ...])
# NOTE the step names can be arbitrary

# Step 1 is what we discussed before - filling the missing values if any using␣
 ↪mean
# Step 2 is feature scaling via standardization - making features look like␣
 ↪normal-distributed
# see sandardization: https://scikit-learn.org/stable/modules/generated/sklearn.
 ↪preprocessing.StandardScaler.html)
num_pipeline = Pipeline(
    steps=[
        ('num_imputer', SimpleImputer()),  # we will tune differet strategies␣
 ↪later
        ('scaler', StandardScaler()),
        ]
)

# Create the preprocessing pipelines for the categorical features
# There are two steps in this pipeline:
# Step 1: filling the missing values if any using the most frequent value
# Step 2: one hot encoding

cat_pipeline = Pipeline(
    steps=[
        ('cat_imputer', SimpleImputer(strategy='most_frequent')),
        ('onehot', OneHotEncoder()),
    ]
)

# Assign features to the pipelines and Combine two pipelines to form the␣
 ↪preprocessor
from sklearn.compose import ColumnTransformer

preprocessor = ColumnTransformer(
    transformers=[
        ('num_pipeline', num_pipeline, num_features),
        ('cat_pipeline', cat_pipeline, cat_features),
    ]
)
```

[12]:
```
# Specify the model to use, which is DecisionTreeClassifier
# Make a full pipeline by combining preprocessor and the model
from sklearn.tree import DecisionTreeClassifier

pipeline_dt = Pipeline(
    steps=[
        ('preprocessor', preprocessor),
        ('clf_dt', DecisionTreeClassifier()),
    ]
)
```

0.4 Decision Tree Model

[13]:
```
# we show how to use GridSearch with K-fold cross validation (K=10) to fine␣
↪tune the model
# we use the accuracy as the scoring metric with training score␣
↪return_train_score=True
from sklearn.model_selection import GridSearchCV

# set up the values of hyperparameters you want to evaluate
# here you must use the step names as the prefix followed by two under_scores␣
↪to sepecify the parameter names and the "full path" of the steps

# we are trying 2 different impputer strategies
# 2x5 different decision tree models with different parameters
# in total we are trying 2x2x1 = 4 different combinations
param_grid_dt = [
    {
        'preprocessor__num_pipeline__num_imputer__strategy': ['mean', 'median'],
        'clf_dt__criterion': ['gini', 'entropy'],
        'clf_dt__max_depth': [10],
    }
]

# set up the grid search
grid_search_dt = GridSearchCV(pipeline_dt, param_grid_dt, cv=10,␣
↪scoring='accuracy')
```

[14]:
```
# train the model using the full pipeline
grid_search_dt.fit(X_train, y_train)
```

[14]:
```
GridSearchCV(cv=10,
             estimator=Pipeline(steps=[('preprocessor',
ColumnTransformer(transformers=[('num_pipeline',
Pipeline(steps=[('num_imputer',
          SimpleImputer()),
         ('scaler',
```

```
          StandardScaler())]),
['Available '
                                                                          'Extra
'
                                                                          'Rooms
'
                                                                          'in '
'Hospital',
'Admission_Deposit',
'Visitors '
                                                                          'with
'
'Patient']),
('cat_pipeline',
Pipeline(steps=[('cat_imputer',
          SimpleImputer(strategy='mos…
'Ward_Type',
'City_Code_Patient',
                                                                          'Type
'
                                                                          'of '
'Admission',
'Severity '
                                                                          'of '
'Illness',
                                                                          'Age',
                                                                          'Bed '
'Grade',
'Hospital_type_code',
'Ward_Facility_Code'])])),
                                      ('clf_dt', DecisionTreeClassifier())]),
             param_grid=[{'clf_dt__criterion': ['gini', 'entropy'],
                          'clf_dt__max_depth': [10],
                          'preprocessor__num_pipeline__num_imputer__strategy':
['mean',
'median']}],
             scoring='accuracy')
```

[15]:
```
# check the best performing parameter combination
grid_search_dt.best_params_
```

[15]:
```
{'clf_dt__criterion': 'gini',
 'clf_dt__max_depth': 10,
 'preprocessor__num_pipeline__num_imputer__strategy': 'median'}
```

[16]:
```
# test score for the 20 decision tree models
grid_search_dt.cv_results_['mean_test_score']
```

[16]: array([0.40811776, 0.40817664, 0.40796467, 0.40793719])

[17]:
```
# best decistion tree model test score
print(grid_search_dt.best_score_)
```

0.40817664376840035

0.5 Random Forest Model

[19]:
```
# try random forest classifer
from sklearn.ensemble import RandomForestClassifier

# rf pipeline
pipeline_rf = Pipeline([
    ('preprocessor', preprocessor),
    ('clf_rf', RandomForestClassifier()),
])

# here we are trying 2x3 different rf models
param_grid_rf = [
    {
        'clf_rf__criterion': ['gini', 'entropy'],
        'clf_rf__n_estimators': [50, 100],
    }
]

# set up the grid search
grid_search_rf = GridSearchCV(pipeline_rf, param_grid_rf, cv=10,␣
 ↪scoring='accuracy', n_jobs=-1)
```

[20]:
```
%%time
# train the model using the full pipeline
grid_search_rf.fit(X_train, y_train)
```

Wall time: 1h 30min 23s

[20]:
```
GridSearchCV(cv=10,
             estimator=Pipeline(steps=[('preprocessor',
ColumnTransformer(transformers=[('num_pipeline',
Pipeline(steps=[('num_imputer',
          SimpleImputer()),
         ('scaler',
          StandardScaler())]),
['Available '
                                                                                    'Extra
'
                                                                                    'Rooms
'
```

```
                                                                          'in '
'Hospital',
'Admission_Deposit',
'Visitors '
                                                                          'with
'
'Patient']),
('cat_pipeline',
Pipeline(steps=[('cat_imputer',
          SimpleImputer(strategy='mos…
          OneHotEncoder())]),
['Hospital_code',
'City_Code_Hospital',
'Department',
'Ward_Type',
'City_Code_Patient',
                                                                          'Type
'
                                                                          'of '
'Admission',
'Severity '
                                                                          'of '
'Illness',
                                                                          'Age',
                                                                          'Bed '
'Grade',
'Hospital_type_code',
'Ward_Facility_Code'])])),
                                       ('clf_rf', RandomForestClassifier())]),
             n_jobs=-1,
             param_grid=[{'clf_rf__criterion': ['gini', 'entropy'],
                          'clf_rf__n_estimators': [50, 100]}],
             scoring='accuracy')
```

[32]:
```
grid_search_rf.best_params_
```

[32]:
```
{'clf_rf__criterion': 'gini', 'clf_rf__n_estimators': 100}
```

[21]:
```
print('best rf score is: ', grid_search_rf.best_score_)
```

```
best rf score is:  0.37430029440628065
```

0.5.1 Logistic Regression implemenation

[15]:
```
df['patient_city_code'] = df['City_Code_Patient'].apply(lambda x: 8 if pd.
 ↪isnull(x) else x)
df['bed_grade'] = df['Bed Grade'].apply(lambda x: 3 if pd.isnull(x) else x)
```

[16]:
```
print(df.dropna().shape)
```

```
(313793, 20)
```

[17]:
```
df['stay'] = df['Stay'].apply(lambda x: '11-20' if x == '20-Nov' else x)
df['stay'] = df['stay'].apply(lambda x: '>100' if x == 'More than 100 Days'␣
 ↪else x)
df.stay[46]
```

[17]:
```
'0-10'
```

Feature Engineering

[18]:
```
le = preprocessing.LabelEncoder()
```

[19]:
```
department_encoded=le.fit_transform(df.Department)
age_encoded = le.fit_transform(df.Age)
admission_encoded=le.fit_transform(df['Type of Admission'])
stay_encoded = le.fit_transform(df.stay)
severity_encoded = le.fit_transform(df['Severity of Illness'])
ward_type_encoded = le.fit_transform(df['Ward_Type'])
ward_facility_encoded = le.fit_transform(df['Ward_Facility_Code'])
hospital_region_encoded = le.fit_transform(df['Hospital_region_code'])
hospital_type_encoded = le.fit_transform(df['Hospital_type_code'])
```

[20]:
```
df['department_encoded'] = department_encoded
df['age_encoded'] = age_encoded
df['admission_encoded'] = admission_encoded
df['stay_encoded'] = stay_encoded
df['severity_encoded'] = severity_encoded
df['ward_type_encoded'] = ward_type_encoded
df['ward_facility_encoded'] = ward_facility_encoded
df['hospital_region_encoded'] = hospital_region_encoded
df['hospital_type_encoded'] = hospital_type_encoded
```

[21]:
```
X = df[['Visitors with Patient','Admission_Deposit','Available Extra Rooms in␣
 ↪Hospital','age_encoded','admission_encoded',␣
 ↪'severity_encoded','Hospital_code','hospital_type_encoded','ward_type_encoded',␣
 ↪'ward_facility_encoded', 'patientid']]
y = df['Stay']
```

[22]:
```
num_features = ['Visitors with Patient','Admission_Deposit','Available Extra␣
 ↪Rooms in Hospital','visit_count','patientid']
cat_features = ['age_encoded','admission_encoded',␣
 ↪'severity_encoded','Hospital_code','hospital_type_encoded','ward_type_encoded',␣
 ↪'ward_facility_encoded']
```

[23]:
```
visit_count = pd.DataFrame(X['patientid'].groupby(X.patientid).agg('count').
 ↪reset_index(name="visit_count"))
visit_count.columns
# visit_count
```

[23]:
```
Index(['patientid', 'visit_count'], dtype='object')
```

[24]:
```
X = X.join(visit_count.set_index('patientid'), lsuffix='_caller',␣
 ↪on='patientid', sort='true')
X.columns
```

[24]:
```
Index(['Visitors with Patient', 'Admission_Deposit',
       'Available Extra Rooms in Hospital', 'age_encoded', 'admission_encoded',
       'severity_encoded', 'Hospital_code', 'hospital_type_encoded',
       'ward_type_encoded', 'ward_facility_encoded', 'patientid',
       'visit_count'],
      dtype='object')
```

[25]:
```
X['visit_count'] = X['visit_count'].fillna(0)
X['visit_count']
```

[25]:
```
254952    4
254953    4
254954    4
254955    4
71206     2
         ..
270876    2
270877    2
199911    3
199912    3
199913    3
Name: visit_count, Length: 318438, dtype: int64
```

Logistic Regression Model pipeline

[26]:
```
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder


num_pipeline = Pipeline(
    steps=[
#          ('num_imputer', SimpleImputer()),  # we will tune differet strategies␣
 ↪later
        ('scaler', StandardScaler()),
        ]
)
```

```
cat_pipeline = Pipeline(
    steps=[
        ('cat_imputer', SimpleImputer(strategy='most_frequent')),
#         ('onehot', OneHotEncoder()),
        ('scaler', StandardScaler()),
    ]
)

# Assign features to the pipelines and Combine two pipelines to form the␣
 ↪preprocessor
from sklearn.compose import ColumnTransformer

preprocessor = ColumnTransformer(
    transformers=[
        ('num_pipeline', num_pipeline, num_features),
        ('cat_pipeline', cat_pipeline, cat_features),
    ]
)
```

[27]:
```
import numpy as np
import matplotlib.pyplot as plt

from sklearn import linear_model, decomposition
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV

logistic = linear_model.LogisticRegression()

pca = decomposition.PCA()
pipe = Pipeline(steps=[
        ('preprocessor', preprocessor),
#         ('pca', pca),
        ('logistic', logistic)])
```

[28]:
```
# This method will allow us to know how long running the model takes,
# each time it runs.
import datetime
from datetime import timedelta

def timediff(s1,s2):
    datetimeFormat = '%H:%M:%S'
    diff = datetime.datetime.strptime(str(s2), datetimeFormat)\
        - datetime.datetime.strptime(str(s1), datetimeFormat)
    return diff
```

[29]:
```
param_grid = [
    {
    'logistic__penalty' : ['l1','l2', 'elasticnet', 'none'],
    'logistic__C' : np.logspace(-4, 4, 20),
    'logistic__solver' : ['lbfgs','newton-cg','liblinear','sag','saga'],
    'logistic__max_iter' : [100, 1000,2500, 5000]
    }
]

param_grid1 = [
    {
    'logistic__penalty' : ['l2','none'],
    'logistic__solver' : ['lbfgs','newton-cg','liblinear','sag','saga'],
    'logistic__max_iter' : [100, 200]
    }
]

param_grid2 = [
    {
    'logistic__penalty' : ['l1'],
    'logistic__solver' : ['liblinear','saga'],
    'logistic__max_iter' : [100, 200]
    }
]

param_grid3 = [
    {
    'logistic__penalty' : ['elasticnet'],
    'logistic__solver' : ['saga'],
    'logistic__max_iter' : [100, 200]
    }
]
```

[30]:
```
s1 = datetime.datetime.now().time().strftime('%H:%M:%S') # START
print(s1)
estimator = GridSearchCV(pipe,param_grid2,cv=3, scoring='accuracy',␣
 ↪verbose=True, n_jobs=-1)
estimator.fit(X, y)
s2 = datetime.datetime.now().time().strftime('%H:%M:%S') # FINISH
print(s2)
print("\n\nElapsed time (HH:MM:SS): ", timediff(s1,s2))
```

```
23:06:59
Fitting 3 folds for each of 4 candidates, totalling 12 fits
23:07:13


Elapsed time (HH:MM:SS):  0:00:14
```

Model evaluation

[31]:
```
estimator.best_params_
```

[31]:
```
{'logistic__max_iter': 100,
 'logistic__penalty': 'l1',
 'logistic__solver': 'liblinear'}
```

[32]:
```
sorted(estimator.cv_results_.keys())
```

[32]:
```
['mean_fit_time',
 'mean_score_time',
 'mean_test_score',
 'param_logistic__max_iter',
 'param_logistic__penalty',
 'param_logistic__solver',
 'params',
 'rank_test_score',
 'split0_test_score',
 'split1_test_score',
 'split2_test_score',
 'std_fit_time',
 'std_score_time',
 'std_test_score']
```

[33]:
```
estimator.cv_results_['mean_test_score']
```

[33]:
```
array([0.2649778, 0.2649778, 0.2649778, 0.2649778])
```

[34]:
```
estimator.best_score_
```

[34]:
```
0.26497779787588166
```

[35]:
```
print("tuned hpyerparameters :(best parameters) ",estimator.best_params_)
print (f'Accuracy - : {estimator.best_score_:.3f}')
```

```
tuned hpyerparameters :(best parameters)  {'logistic__max_iter': 100,
'logistic__penalty': 'l1', 'logistic__solver': 'liblinear'}
Accuracy - : 0.265
```

[36]:
```
clf_best = estimator.best_estimator_
clf_best
```

[36]:
```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num_pipeline',
                                                  Pipeline(steps=[('scaler',
StandardScaler())]),
                                                  ['Visitors with Patient',
                                                   'Admission_Deposit',
                                                   'Available Extra Rooms in '
```

```
                                                     'Hospital',
                                                     'visit_count',
                                                     'patientid']),
                                                   ('cat_pipeline',
Pipeline(steps=[('cat_imputer',
SimpleImputer(strategy='most_frequent')),
                                                                   ('scaler',
StandardScaler())]),
                                                    ['age_encoded',
                                                     'admission_encoded',
                                                     'severity_encoded',
                                                     'Hospital_code',
                                                     'hospital_type_encoded',
                                                     'ward_type_encoded',
                                                     'ward_facility_encoded'])])),
                ('logistic',
                 LogisticRegression(penalty='l1', solver='liblinear'))])
```

[37]:
```
clf_best.named_steps
```

[37]:
```
{'preprocessor': ColumnTransformer(transformers=[('num_pipeline',
                                  Pipeline(steps=[('scaler',
StandardScaler())]),
                                  ['Visitors with Patient', 'Admission_Deposit',
                                   'Available Extra Rooms in Hospital',
                                   'visit_count', 'patientid']),
                                 ('cat_pipeline',
                                  Pipeline(steps=[('cat_imputer',
SimpleImputer(strategy='most_frequent')),
                                                  ('scaler',
StandardScaler())]),
                                  ['age_encoded', 'admission_encoded',
                                   'severity_encoded', 'Hospital_code',
                                   'hospital_type_encoded', 'ward_type_encoded',
                                   'ward_facility_encoded'])]),
 'logistic': LogisticRegression(penalty='l1', solver='liblinear')}
```

[38]:
```
clf_best['logistic']
```

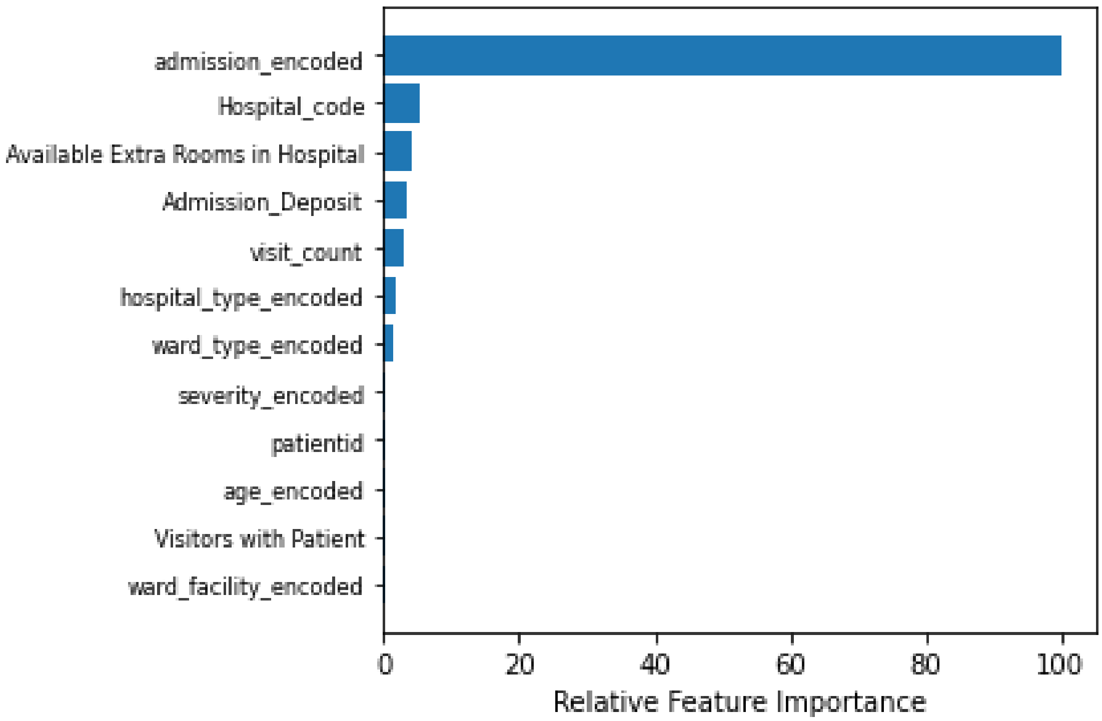
[38]:
```
LogisticRegression(penalty='l1', solver='liblinear')
```

[39]:
```
feature_importance = abs(clf_best['logistic'].coef_[0])
feature_importance = 100.0 * (feature_importance / feature_importance.max())
sorted_idx = np.argsort(feature_importance)
pos = np.arange(sorted_idx.shape[0]) + .5

featfig = plt.figure()
```

```
featax = featfig.add_subplot(1, 1, 1)
featax.barh(pos, feature_importance[sorted_idx], align='center')
featax.set_yticks(pos)
featax.set_yticklabels(np.array(X.columns)[sorted_idx], fontsize=8)
featax.set_xlabel('Relative Feature Importance')

plt.tight_layout()
plt.show()
```



0.6 Model Best Estimator

[20]:
```
clf_best = grid_search_dt.best_estimator_
clf_best
```

[20]:
```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num_pipeline',
Pipeline(steps=[('num_imputer',
SimpleImputer(strategy='median')),
                                                                   ('scaler',
StandardScaler())]),
                                                  ['Available Extra Rooms in '
                                                   'Hospital',
                                                   'Admission_Deposit',
```

```
                                                    'Visitors with Patient']),
                                                  ('cat_pipeline',
Pipeline(steps=[('cat_imputer',
SimpleImputer(strategy='most_frequent')),
                                                                     ('onehot',
OneHotEncoder())]),
                                                   ['Hospital_code',
                                                    'City_Code_Hospital',
                                                    'Department', 'Ward_Type',
                                                    'City_Code_Patient',
                                                    'Type of Admission',
                                                    'Severity of Illness', 'Age',
                                                    'Bed Grade',
                                                    'Hospital_type_code',
                                                    'Ward_Facility_Code'])])),
                ('clf_dt', DecisionTreeClassifier(max_depth=10))])
```

0.7 Baseline Prediction

[22]:
```python
baseline = pd.DataFrame(df.patientid.groupby([df.Stay]).agg('count').
 ↪reset_index(name="count"))
baseline['percent'] = (baseline['count'] / baseline['count'].sum()) * 100
baseline
```

[22]:

	Stay	count	percent
0	0-10	23604	7.412432
1	11-20	78139	24.538215
2	21-30	87491	27.475050
3	31-40	55159	17.321739
4	41-50	11743	3.687688
5	51-60	35018	10.996803
6	61-70	2744	0.861706
7	71-80	10254	3.220093
8	81-90	4838	1.519291
9	91-100	2765	0.868301
10	More than 100 Days	6683	2.098682

[31]:
```python
print(f'Baseline Accuracy Percentage: {max(baseline.percent)}')
print('Best RF Accuracy Percentage: ', grid_search_rf.best_score_*100)
print('Best DT Accuracy Percentage: ', grid_search_dt.best_score_*100)
```

```
Baseline Accuracy Percentage: 27.475050088243236
Best RF Accuracy Percentage:  37.43002944062807
Best DT Accuracy Percentage:  40.817664376840035
```

0.8 Feature Importance

[21]:
```
clf_best.named_steps
```

[21]:
```
{'preprocessor': ColumnTransformer(transformers=[('num_pipeline',
                                  Pipeline(steps=[('num_imputer',
SimpleImputer(strategy='median')),
                                                  ('scaler',
StandardScaler())]),
                                  ['Available Extra Rooms in Hospital',
                                   'Admission_Deposit',
                                   'Visitors with Patient']),
                                 ('cat_pipeline',
                                  Pipeline(steps=[('cat_imputer',
SimpleImputer(strategy='most_frequent')),
                                                  ('onehot', OneHotEncoder())]),
                                  ['Hospital_code', 'City_Code_Hospital',
                                   'Department', 'Ward_Type',
                                   'City_Code_Patient', 'Type of Admission',
                                   'Severity of Illness', 'Age', 'Bed Grade',
                                   'Hospital_type_code',
                                   'Ward_Facility_Code'])]),
 'clf_dt': DecisionTreeClassifier(max_depth=10)}
```

[22]:
```
clf_best.named_steps['preprocessor']
```

[22]:
```
ColumnTransformer(transformers=[('num_pipeline',
                                 Pipeline(steps=[('num_imputer',
SimpleImputer(strategy='median')),
                                                 ('scaler', StandardScaler())]),
                                 ['Available Extra Rooms in Hospital',
                                  'Admission_Deposit',
                                  'Visitors with Patient']),
                                ('cat_pipeline',
                                 Pipeline(steps=[('cat_imputer',
SimpleImputer(strategy='most_frequent')),
                                                 ('onehot', OneHotEncoder())]),
                                 ['Hospital_code', 'City_Code_Hospital',
                                  'Department', 'Ward_Type',
                                  'City_Code_Patient', 'Type of Admission',
                                  'Severity of Illness', 'Age', 'Bed Grade',
                                  'Hospital_type_code',
                                  'Ward_Facility_Code'])])
```

[23]:
```
i = clf_best['clf_dt'].feature_importances_
i
```

[23]:
```
array([1.11020387e-02, 6.50303562e-02, 4.46830404e-01, 2.97695088e-04,
       0.00000000e+00, 6.14074094e-05, 9.44451918e-05, 0.00000000e+00,
       3.04280661e-04, 0.00000000e+00, 1.35284254e-04, 5.86680334e-05,
       5.50456826e-04, 4.34496297e-04, 2.59477091e-04, 6.08913980e-04,
       1.35513324e-03, 0.00000000e+00, 2.16784521e-04, 1.78066048e-03,
       9.44940374e-04, 2.13032275e-02, 7.01590999e-05, 0.00000000e+00,
       0.00000000e+00, 1.69774191e-04, 3.32691370e-03, 1.33428755e-04,
       5.43829356e-03, 5.22757079e-04, 1.35674609e-04, 7.80417935e-05,
       9.17620452e-04, 0.00000000e+00, 1.90892193e-04, 5.27654905e-04,
       7.58049488e-03, 2.46613116e-04, 9.31755294e-05, 2.71841239e-04,
       4.40552079e-04, 1.02592359e-02, 5.75474897e-04, 1.01129269e-04,
       9.45625136e-05, 0.00000000e+00, 2.15517520e-04, 1.02193172e-03,
       2.10912216e-03, 1.09315840e-03, 1.02553491e-04, 3.35730050e-02,
       1.33119906e-01, 9.11969229e-04, 4.04569377e-02, 0.00000000e+00,
       0.00000000e+00, 8.93641050e-04, 2.62254631e-03, 2.30080110e-04,
       1.35478942e-03, 6.43645888e-04, 2.99622879e-04, 1.00758443e-03,
       1.70843074e-02, 5.38976324e-04, 7.19981647e-04, 0.00000000e+00,
       3.50727985e-04, 9.87944271e-04, 2.01852958e-04, 3.39335216e-04,
       2.47817762e-04, 1.01127801e-04, 2.52913220e-05, 0.00000000e+00,
       1.70128165e-04, 0.00000000e+00, 3.69033617e-04, 0.00000000e+00,
       6.06991727e-05, 4.38382914e-05, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 4.36473884e-02, 1.17385047e-02,
       1.61997645e-03, 4.90982346e-03, 1.60706630e-02, 7.71220309e-04,
       2.75838339e-04, 8.37794018e-04, 5.08260540e-04, 5.18230963e-04,
       8.45248424e-04, 1.50174153e-03, 1.31891407e-03, 9.44741297e-04,
       1.97851444e-04, 3.71562621e-04, 2.19277415e-02, 5.11778393e-02,
       7.88154342e-04, 1.77283766e-04, 1.95221421e-03, 2.03877148e-03,
       4.64240948e-04, 0.00000000e+00, 8.97900884e-05, 2.32899452e-04,
       1.19042604e-04, 7.21326959e-04, 8.63906207e-05, 6.59895149e-03,
       2.82832274e-03, 9.56203713e-04, 3.23034105e-04])
```

[24]:
```python
clf_best['preprocessor'].transformers_
```

[24]:
```
[('num_pipeline',
  Pipeline(steps=[('num_imputer', SimpleImputer(strategy='median')),
                  ('scaler', StandardScaler())]),
  ['Available Extra Rooms in Hospital',
   'Admission_Deposit',
   'Visitors with Patient']),
 ('cat_pipeline',
  Pipeline(steps=[('cat_imputer', SimpleImputer(strategy='most_frequent')),
                  ('onehot', OneHotEncoder())]),
  ['Hospital_code',
   'City_Code_Hospital',
   'Department',
```

```
    'Ward_Type',
    'City_Code_Patient',
    'Type of Admission',
    'Severity of Illness',
    'Age',
    'Bed Grade',
    'Hospital_type_code',
    'Ward_Facility_Code']),
  ('remainder', 'drop', [3, 9])]
```

[25]:
```
# get columnTransformer
clf_best[0]
```

[25]:
```
ColumnTransformer(transformers=[('num_pipeline',
                                 Pipeline(steps=[('num_imputer',
SimpleImputer(strategy='median')),
                                                 ('scaler', StandardScaler())]),
                                 ['Available Extra Rooms in Hospital',
                                  'Admission_Deposit',
                                  'Visitors with Patient']),
                                ('cat_pipeline',
                                 Pipeline(steps=[('cat_imputer',
SimpleImputer(strategy='most_frequent')),
                                                 ('onehot', OneHotEncoder())]),
                                 ['Hospital_code', 'City_Code_Hospital',
                                  'Department', 'Ward_Type',
                                  'City_Code_Patient', 'Type of Admission',
                                  'Severity of Illness', 'Age', 'Bed Grade',
                                  'Hospital_type_code',
                                  'Ward_Facility_Code'])])
```

[26]:
```
clf_best[0].transformers_
```

[26]:
```
[('num_pipeline',
  Pipeline(steps=[('num_imputer', SimpleImputer(strategy='median')),
                  ('scaler', StandardScaler())]),
  ['Available Extra Rooms in Hospital',
   'Admission_Deposit',
   'Visitors with Patient']),
 ('cat_pipeline',
  Pipeline(steps=[('cat_imputer', SimpleImputer(strategy='most_frequent')),
                  ('onehot', OneHotEncoder())]),
  ['Hospital_code',
   'City_Code_Hospital',
   'Department',
   'Ward_Type',
   'City_Code_Patient',
```

```
  'Type of Admission',
  'Severity of Illness',
  'Age',
  'Bed Grade',
  'Hospital_type_code',
  'Ward_Facility_Code']),
('remainder', 'drop', [3, 9])]
```

[27]:
```python
num_original_feature_names = clf_best[0].transformers_[0][2]
num_original_feature_names
```

[27]:
```
['Available Extra Rooms in Hospital',
 'Admission_Deposit',
 'Visitors with Patient']
```

[28]:
```python
cat_original_feature_names = clf_best[0].transformers_[1][2]
cat_original_feature_names
```

[28]:
```
['Hospital_code',
 'City_Code_Hospital',
 'Department',
 'Ward_Type',
 'City_Code_Patient',
 'Type of Admission',
 'Severity of Illness',
 'Age',
 'Bed Grade',
 'Hospital_type_code',
 'Ward_Facility_Code']
```

[29]:
```python
cat_new_feature_names = list(clf_best[0].transformers_[1][1]['onehot'].
 ↪get_feature_names(cat_original_feature_names))
cat_new_feature_names
```

[29]:
```
['Hospital_code_1',
 'Hospital_code_2',
 'Hospital_code_3',
 'Hospital_code_4',
 'Hospital_code_5',
 'Hospital_code_6',
 'Hospital_code_7',
 'Hospital_code_8',
 'Hospital_code_9',
 'Hospital_code_10',
 'Hospital_code_11',
 'Hospital_code_12',
 'Hospital_code_13',
```

```
'Hospital_code_14',
'Hospital_code_15',
'Hospital_code_16',
'Hospital_code_17',
'Hospital_code_18',
'Hospital_code_19',
'Hospital_code_20',
'Hospital_code_21',
'Hospital_code_22',
'Hospital_code_23',
'Hospital_code_24',
'Hospital_code_25',
'Hospital_code_26',
'Hospital_code_27',
'Hospital_code_28',
'Hospital_code_29',
'Hospital_code_30',
'Hospital_code_31',
'Hospital_code_32',
'City_Code_Hospital_1',
'City_Code_Hospital_2',
'City_Code_Hospital_3',
'City_Code_Hospital_4',
'City_Code_Hospital_5',
'City_Code_Hospital_6',
'City_Code_Hospital_7',
'City_Code_Hospital_9',
'City_Code_Hospital_10',
'City_Code_Hospital_11',
'City_Code_Hospital_13',
'Department_TB & Chest disease',
'Department_anesthesia',
'Department_gynecology',
'Department_radiotherapy',
'Department_surgery',
'Ward_Type_P',
'Ward_Type_Q',
'Ward_Type_R',
'Ward_Type_S',
'Ward_Type_T',
'Ward_Type_U',
'City_Code_Patient_1.0',
'City_Code_Patient_2.0',
'City_Code_Patient_3.0',
'City_Code_Patient_4.0',
'City_Code_Patient_5.0',
'City_Code_Patient_6.0',
```

```
'City_Code_Patient_7.0',
'City_Code_Patient_8.0',
'City_Code_Patient_9.0',
'City_Code_Patient_10.0',
'City_Code_Patient_11.0',
'City_Code_Patient_12.0',
'City_Code_Patient_13.0',
'City_Code_Patient_14.0',
'City_Code_Patient_15.0',
'City_Code_Patient_16.0',
'City_Code_Patient_18.0',
'City_Code_Patient_19.0',
'City_Code_Patient_20.0',
'City_Code_Patient_21.0',
'City_Code_Patient_22.0',
'City_Code_Patient_23.0',
'City_Code_Patient_24.0',
'City_Code_Patient_25.0',
'City_Code_Patient_26.0',
'City_Code_Patient_27.0',
'City_Code_Patient_28.0',
'City_Code_Patient_29.0',
'City_Code_Patient_30.0',
'City_Code_Patient_31.0',
'City_Code_Patient_32.0',
'City_Code_Patient_33.0',
'City_Code_Patient_34.0',
'City_Code_Patient_35.0',
'City_Code_Patient_36.0',
'City_Code_Patient_37.0',
'City_Code_Patient_38.0',
'Type of Admission_Emergency',
'Type of Admission_Trauma',
'Type of Admission_Urgent',
'Severity of Illness_Extreme',
'Severity of Illness_Minor',
'Severity of Illness_Moderate',
'Age_0-10',
'Age_11-20',
'Age_21-30',
'Age_31-40',
'Age_41-50',
'Age_51-60',
'Age_61-70',
'Age_71-80',
'Age_81-90',
'Age_91-100',
```

```
 'Bed Grade_1.0',
 'Bed Grade_2.0',
 'Bed Grade_3.0',
 'Bed Grade_4.0',
 'Hospital_type_code_a',
 'Hospital_type_code_b',
 'Hospital_type_code_c',
 'Hospital_type_code_d',
 'Hospital_type_code_e',
 'Hospital_type_code_f',
 'Hospital_type_code_g',
 'Ward_Facility_Code_A',
 'Ward_Facility_Code_B',
 'Ward_Facility_Code_C',
 'Ward_Facility_Code_D',
 'Ward_Facility_Code_E',
 'Ward_Facility_Code_F']
```

[30]:
```
feature_names = num_original_feature_names + cat_new_feature_names
feature_names
```

[30]:
```
['Available Extra Rooms in Hospital',
 'Admission_Deposit',
 'Visitors with Patient',
 'Hospital_code_1',
 'Hospital_code_2',
 'Hospital_code_3',
 'Hospital_code_4',
 'Hospital_code_5',
 'Hospital_code_6',
 'Hospital_code_7',
 'Hospital_code_8',
 'Hospital_code_9',
 'Hospital_code_10',
 'Hospital_code_11',
 'Hospital_code_12',
 'Hospital_code_13',
 'Hospital_code_14',
 'Hospital_code_15',
 'Hospital_code_16',
 'Hospital_code_17',
 'Hospital_code_18',
 'Hospital_code_19',
 'Hospital_code_20',
 'Hospital_code_21',
 'Hospital_code_22',
 'Hospital_code_23',
```

```
'Hospital_code_24',
'Hospital_code_25',
'Hospital_code_26',
'Hospital_code_27',
'Hospital_code_28',
'Hospital_code_29',
'Hospital_code_30',
'Hospital_code_31',
'Hospital_code_32',
'City_Code_Hospital_1',
'City_Code_Hospital_2',
'City_Code_Hospital_3',
'City_Code_Hospital_4',
'City_Code_Hospital_5',
'City_Code_Hospital_6',
'City_Code_Hospital_7',
'City_Code_Hospital_9',
'City_Code_Hospital_10',
'City_Code_Hospital_11',
'City_Code_Hospital_13',
'Department_TB & Chest disease',
'Department_anesthesia',
'Department_gynecology',
'Department_radiotherapy',
'Department_surgery',
'Ward_Type_P',
'Ward_Type_Q',
'Ward_Type_R',
'Ward_Type_S',
'Ward_Type_T',
'Ward_Type_U',
'City_Code_Patient_1.0',
'City_Code_Patient_2.0',
'City_Code_Patient_3.0',
'City_Code_Patient_4.0',
'City_Code_Patient_5.0',
'City_Code_Patient_6.0',
'City_Code_Patient_7.0',
'City_Code_Patient_8.0',
'City_Code_Patient_9.0',
'City_Code_Patient_10.0',
'City_Code_Patient_11.0',
'City_Code_Patient_12.0',
'City_Code_Patient_13.0',
'City_Code_Patient_14.0',
'City_Code_Patient_15.0',
'City_Code_Patient_16.0',
```

```
'City_Code_Patient_18.0',
'City_Code_Patient_19.0',
'City_Code_Patient_20.0',
'City_Code_Patient_21.0',
'City_Code_Patient_22.0',
'City_Code_Patient_23.0',
'City_Code_Patient_24.0',
'City_Code_Patient_25.0',
'City_Code_Patient_26.0',
'City_Code_Patient_27.0',
'City_Code_Patient_28.0',
'City_Code_Patient_29.0',
'City_Code_Patient_30.0',
'City_Code_Patient_31.0',
'City_Code_Patient_32.0',
'City_Code_Patient_33.0',
'City_Code_Patient_34.0',
'City_Code_Patient_35.0',
'City_Code_Patient_36.0',
'City_Code_Patient_37.0',
'City_Code_Patient_38.0',
'Type of Admission_Emergency',
'Type of Admission_Trauma',
'Type of Admission_Urgent',
'Severity of Illness_Extreme',
'Severity of Illness_Minor',
'Severity of Illness_Moderate',
'Age_0-10',
'Age_11-20',
'Age_21-30',
'Age_31-40',
'Age_41-50',
'Age_51-60',
'Age_61-70',
'Age_71-80',
'Age_81-90',
'Age_91-100',
'Bed Grade_1.0',
'Bed Grade_2.0',
'Bed Grade_3.0',
'Bed Grade_4.0',
'Hospital_type_code_a',
'Hospital_type_code_b',
'Hospital_type_code_c',
'Hospital_type_code_d',
'Hospital_type_code_e',
'Hospital_type_code_f',
```

```
 'Hospital_type_code_g',
 'Ward_Facility_Code_A',
 'Ward_Facility_Code_B',
 'Ward_Facility_Code_C',
 'Ward_Facility_Code_D',
 'Ward_Facility_Code_E',
 'Ward_Facility_Code_F']
```

[31]:
```
r = pd.DataFrame(i, index=feature_names, columns=['importance'])
pd.set_option('display.max_rows', None)
r
```

[31]:

	importance
Available Extra Rooms in Hospital	0.011102
Admission_Deposit	0.065030
Visitors with Patient	0.446830
Hospital_code_1	0.000298
Hospital_code_2	0.000000
Hospital_code_3	0.000061
Hospital_code_4	0.000094
Hospital_code_5	0.000000
Hospital_code_6	0.000304
Hospital_code_7	0.000000
Hospital_code_8	0.000135
Hospital_code_9	0.000059
Hospital_code_10	0.000550
Hospital_code_11	0.000434
Hospital_code_12	0.000259
Hospital_code_13	0.000609
Hospital_code_14	0.001355
Hospital_code_15	0.000000
Hospital_code_16	0.000217
Hospital_code_17	0.001781
Hospital_code_18	0.000945
Hospital_code_19	0.021303
Hospital_code_20	0.000070
Hospital_code_21	0.000000
Hospital_code_22	0.000000
Hospital_code_23	0.000170
Hospital_code_24	0.003327
Hospital_code_25	0.000133
Hospital_code_26	0.005438
Hospital_code_27	0.000523
Hospital_code_28	0.000136
Hospital_code_29	0.000078
Hospital_code_30	0.000918
Hospital_code_31	0.000000

Hospital_code_32	0.000191
City_Code_Hospital_1	0.000528
City_Code_Hospital_2	0.007580
City_Code_Hospital_3	0.000247
City_Code_Hospital_4	0.000093
City_Code_Hospital_5	0.000272
City_Code_Hospital_6	0.000441
City_Code_Hospital_7	0.010259
City_Code_Hospital_9	0.000575
City_Code_Hospital_10	0.000101
City_Code_Hospital_11	0.000095
City_Code_Hospital_13	0.000000
Department_TB & Chest disease	0.000216
Department_anesthesia	0.001022
Department_gynecology	0.002109
Department_radiotherapy	0.001093
Department_surgery	0.000103
Ward_Type_P	0.033573
Ward_Type_Q	0.133120
Ward_Type_R	0.000912
Ward_Type_S	0.040457
Ward_Type_T	0.000000
Ward_Type_U	0.000000
City_Code_Patient_1.0	0.000894
City_Code_Patient_2.0	0.002623
City_Code_Patient_3.0	0.000230
City_Code_Patient_4.0	0.001355
City_Code_Patient_5.0	0.000644
City_Code_Patient_6.0	0.000300
City_Code_Patient_7.0	0.001008
City_Code_Patient_8.0	0.017084
City_Code_Patient_9.0	0.000539
City_Code_Patient_10.0	0.000720
City_Code_Patient_11.0	0.000000
City_Code_Patient_12.0	0.000351
City_Code_Patient_13.0	0.000988
City_Code_Patient_14.0	0.000202
City_Code_Patient_15.0	0.000339
City_Code_Patient_16.0	0.000248
City_Code_Patient_18.0	0.000101
City_Code_Patient_19.0	0.000025
City_Code_Patient_20.0	0.000000
City_Code_Patient_21.0	0.000170
City_Code_Patient_22.0	0.000000
City_Code_Patient_23.0	0.000369
City_Code_Patient_24.0	0.000000
City_Code_Patient_25.0	0.000061

City_Code_Patient_26.0	0.000044
City_Code_Patient_27.0	0.000000
City_Code_Patient_28.0	0.000000
City_Code_Patient_29.0	0.000000
City_Code_Patient_30.0	0.000000
City_Code_Patient_31.0	0.000000
City_Code_Patient_32.0	0.000000
City_Code_Patient_33.0	0.000000
City_Code_Patient_34.0	0.000000
City_Code_Patient_35.0	0.000000
City_Code_Patient_36.0	0.000000
City_Code_Patient_37.0	0.000000
City_Code_Patient_38.0	0.000000
Type of Admission_Emergency	0.043647
Type of Admission_Trauma	0.011739
Type of Admission_Urgent	0.001620
Severity of Illness_Extreme	0.004910
Severity of Illness_Minor	0.016071
Severity of Illness_Moderate	0.000771
Age_0-10	0.000276
Age_11-20	0.000838
Age_21-30	0.000508
Age_31-40	0.000518
Age_41-50	0.000845
Age_51-60	0.001502
Age_61-70	0.001319
Age_71-80	0.000945
Age_81-90	0.000198
Age_91-100	0.000372
Bed Grade_1.0	0.021928
Bed Grade_2.0	0.051178
Bed Grade_3.0	0.000788
Bed Grade_4.0	0.000177
Hospital_type_code_a	0.001952
Hospital_type_code_b	0.002039
Hospital_type_code_c	0.000464
Hospital_type_code_d	0.000000
Hospital_type_code_e	0.000090
Hospital_type_code_f	0.000233
Hospital_type_code_g	0.000119
Ward_Facility_Code_A	0.000721
Ward_Facility_Code_B	0.000086
Ward_Facility_Code_C	0.006599
Ward_Facility_Code_D	0.002828
Ward_Facility_Code_E	0.000956
Ward_Facility_Code_F	0.000323

[32]:
```
r.sort_values('importance', ascending=False)
```

[32]:

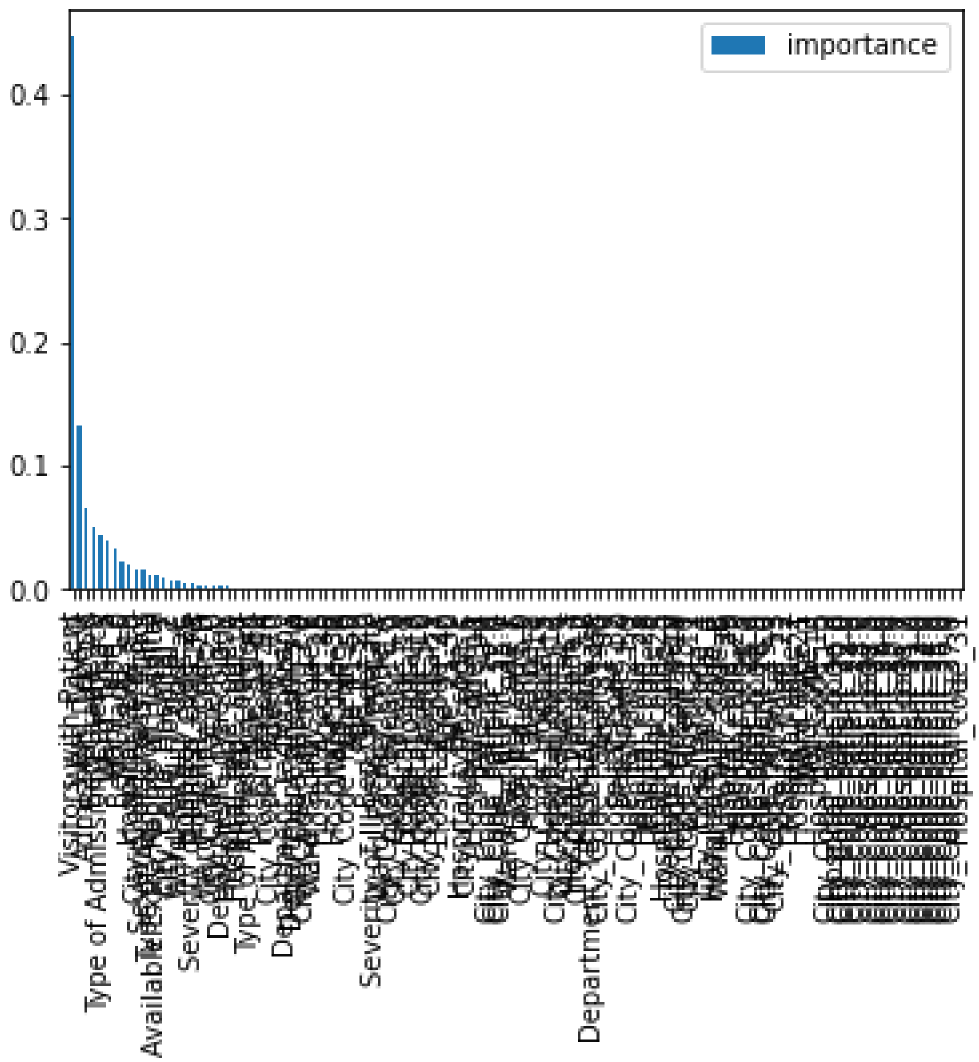
	importance
Visitors with Patient	0.446830
Ward_Type_Q	0.133120
Admission_Deposit	0.065030
Bed Grade_2.0	0.051178
Type of Admission_Emergency	0.043647
Ward_Type_S	0.040457
Ward_Type_P	0.033573
Bed Grade_1.0	0.021928
Hospital_code_19	0.021303
City_Code_Patient_8.0	0.017084
Severity of Illness_Minor	0.016071
Type of Admission_Trauma	0.011739
Available Extra Rooms in Hospital	0.011102
City_Code_Hospital_7	0.010259
City_Code_Hospital_2	0.007580
Ward_Facility_Code_C	0.006599
Hospital_code_26	0.005438
Severity of Illness_Extreme	0.004910
Hospital_code_24	0.003327
Ward_Facility_Code_D	0.002828
City_Code_Patient_2.0	0.002623
Department_gynecology	0.002109
Hospital_type_code_b	0.002039
Hospital_type_code_a	0.001952
Hospital_code_17	0.001781
Type of Admission_Urgent	0.001620
Age_51-60	0.001502
Hospital_code_14	0.001355
City_Code_Patient_4.0	0.001355
Age_61-70	0.001319
Department_radiotherapy	0.001093
Department_anesthesia	0.001022
City_Code_Patient_7.0	0.001008
City_Code_Patient_13.0	0.000988
Ward_Facility_Code_E	0.000956
Hospital_code_18	0.000945
Age_71-80	0.000945
Hospital_code_30	0.000918
Ward_Type_R	0.000912
City_Code_Patient_1.0	0.000894
Age_41-50	0.000845
Age_11-20	0.000838
Bed Grade_3.0	0.000788
Severity of Illness_Moderate	0.000771

Ward_Facility_Code_A	0.000721
City_Code_Patient_10.0	0.000720
City_Code_Patient_5.0	0.000644
Hospital_code_13	0.000609
City_Code_Hospital_9	0.000575
Hospital_code_10	0.000550
City_Code_Patient_9.0	0.000539
City_Code_Hospital_1	0.000528
Hospital_code_27	0.000523
Age_31-40	0.000518
Age_21-30	0.000508
Hospital_type_code_c	0.000464
City_Code_Hospital_6	0.000441
Hospital_code_11	0.000434
Age_91-100	0.000372
City_Code_Patient_23.0	0.000369
City_Code_Patient_12.0	0.000351
City_Code_Patient_15.0	0.000339
Ward_Facility_Code_F	0.000323
Hospital_code_6	0.000304
City_Code_Patient_6.0	0.000300
Hospital_code_1	0.000298
Age_0-10	0.000276
City_Code_Hospital_5	0.000272
Hospital_code_12	0.000259
City_Code_Patient_16.0	0.000248
City_Code_Hospital_3	0.000247
Hospital_type_code_f	0.000233
City_Code_Patient_3.0	0.000230
Hospital_code_16	0.000217
Department_TB & Chest disease	0.000216
City_Code_Patient_14.0	0.000202
Age_81-90	0.000198
Hospital_code_32	0.000191
Bed Grade_4.0	0.000177
City_Code_Patient_21.0	0.000170
Hospital_code_23	0.000170
Hospital_code_28	0.000136
Hospital_code_8	0.000135
Hospital_code_25	0.000133
Hospital_type_code_g	0.000119
Department_surgery	0.000103
City_Code_Hospital_10	0.000101
City_Code_Patient_18.0	0.000101
City_Code_Hospital_11	0.000095
Hospital_code_4	0.000094
City_Code_Hospital_4	0.000093

```
Hospital_type_code_e           0.000090
Ward_Facility_Code_B           0.000086
Hospital_code_29               0.000078
Hospital_code_20               0.000070
Hospital_code_3                0.000061
City_Code_Patient_25.0         0.000061
Hospital_code_9                0.000059
City_Code_Patient_26.0         0.000044
City_Code_Patient_19.0         0.000025
City_Code_Hospital_13          0.000000
Hospital_code_22               0.000000
Hospital_code_2                0.000000
Hospital_code_21               0.000000
Ward_Type_T                    0.000000
Ward_Type_U                    0.000000
Hospital_code_15               0.000000
City_Code_Patient_11.0         0.000000
Hospital_type_code_d           0.000000
City_Code_Patient_20.0         0.000000
City_Code_Patient_22.0         0.000000
City_Code_Patient_24.0         0.000000
City_Code_Patient_38.0         0.000000
Hospital_code_5                0.000000
City_Code_Patient_27.0         0.000000
City_Code_Patient_28.0         0.000000
City_Code_Patient_29.0         0.000000
City_Code_Patient_30.0         0.000000
Hospital_code_7                0.000000
City_Code_Patient_31.0         0.000000
City_Code_Patient_32.0         0.000000
City_Code_Patient_33.0         0.000000
City_Code_Patient_34.0         0.000000
City_Code_Patient_35.0         0.000000
City_Code_Patient_36.0         0.000000
City_Code_Patient_37.0         0.000000
Hospital_code_31               0.000000
```

[33]:
```
r.sort_values('importance', ascending=False).plot.bar()
```

[33]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x22811b1d430>
```



0.9 Persisting the Model

[35]:
```python
# Save the model as a pickle file
import joblib
joblib.dump(clf_best, "2clf-best.pickle")
```

[35]: ['2clf-best.pickle']

[37]:
```python
saved_tree_clf = joblib.load("2clf-best.pickle")
saved_tree_clf
```

[37]:
```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num_pipeline',
Pipeline(steps=[('num_imputer',
```

```
 SimpleImputer(strategy='median')),
                                                                   ('scaler',
 StandardScaler())]),
                                                    ['Available Extra Rooms in '
                                                     'Hospital',
                                                     'Admission_Deposit',
                                                     'Visitors with Patient']),
                                                   ('cat_pipeline',
 Pipeline(steps=[('cat_imputer',
 SimpleImputer(strategy='most_frequent')),
                                                                   ('onehot',
 OneHotEncoder())]),
                                                    ['Hospital_code',
                                                     'City_Code_Hospital',
                                                     'Department', 'Ward_Type',
                                                     'City_Code_Patient',
                                                     'Type of Admission',
                                                     'Severity of Illness', 'Age',
                                                     'Bed Grade',
                                                     'Hospital_type_code',
                                                     'Ward_Facility_Code'])])),
                ('clf_dt', DecisionTreeClassifier(max_depth=10))])
```

[57]:
```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 318438 entries, 0 to 318437
Data columns (total 16 columns):
 #   Column                             Non-Null Count   Dtype
---  ------                             --------------   -----
 0   Hospital_code                      318438 non-null  int64
 1   Hospital_type_code                 318438 non-null  object
 2   City_Code_Hospital                 318438 non-null  int64
 3   Hospital_region_code               318438 non-null  object
 4   Available Extra Rooms in Hospital  318438 non-null  int64
 5   Department                         318438 non-null  object
 6   Ward_Type                          318438 non-null  object
 7   Ward_Facility_Code                 318438 non-null  object
 8   Bed Grade                          318325 non-null  float64
 9   patientid                          318438 non-null  int64
 10  City_Code_Patient                  313906 non-null  float64
 11  Type of Admission                  318438 non-null  object
 12  Severity of Illness                318438 non-null  object
 13  Visitors with Patient              318438 non-null  int64
 14  Age                                318438 non-null  object
 15  Admission_Deposit                  318438 non-null  float64
dtypes: float64(3), int64(5), object(8)
memory usage: 38.9+ MB
```

[58]:
```
patient2 = pd.DataFrame(
    {
        'Hospital_code': [8],
        'Hospital_type_code': ['c'],
        'City_Code_Hospital': [4],
        'Hospital_region_code': ['Z'],
        'Available Extra Rooms in Hospital': [2],
        'Department': ['radiotherapy'],
        'Ward_Type': ['R'],
        'Ward_Facility_Code': ['D'],
        'Bed Grade': [2.0],
        'patientid': [8088],
        'City_Code_Patient': [8.0],
        'Type of Admission': ['Emergency'],
        'Severity of Illness': ['Moderate'],
        'Visitors with Patient': [4],
        'Age': ['31-40'],
        'Admission_Deposit': [4091.0]
    }
)
patient2
```

[58]:
```
   Hospital_code Hospital_type_code  City_Code_Hospital Hospital_region_code  \
0              8                  c                   4                    Z

   Available Extra Rooms in Hospital    Department Ward_Type  \
0                                  2  radiotherapy         R

  Ward_Facility_Code  Bed Grade  patientid  City_Code_Patient  \
0                  D        2.0       8088                8.0

  Type of Admission Severity of Illness  Visitors with Patient    Age  \
0         Emergency            Moderate                      4  31-40

   Admission_Deposit
0             4091.0
```

[59]:
```
pred2 = saved_tree_clf.predict(patient2)
```

[60]:
```
pred2
```

[60]:
```
array(['51-60'], dtype=object)
```