In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
churn_data = pd.read_csv('https://raw.githubusercontent.com/zekelabs/data-science-complete-tutorial/master/Data/churn.csv.txt', parse_dates=['last_trip_date','signup_date'])
```

In [3]:
```
churn_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 12 columns):
avg_dist                50000 non-null float64
avg_rating_by_driver    49799 non-null float64
avg_rating_of_driver    41878 non-null float64
avg_surge               50000 non-null float64
city                    50000 non-null object
last_trip_date          50000 non-null datetime64[ns]
phone                   49604 non-null object
signup_date             50000 non-null datetime64[ns]
surge_pct               50000 non-null float64
trips_in_first_30_days  50000 non-null int64
luxury_car_user         50000 non-null bool
weekday_pct             50000 non-null float64
dtypes: bool(1), datetime64[ns](2), float64(6), int64(1), object(2)
memory usage: 4.2+ MB
```

In [4]:
```
churn_data.last_trip_date.max()
```

Out[4]:
```
Timestamp('2014-07-01 00:00:00')
```

In [5]:
```
import datetime
cutoff = churn_data.last_trip_date.max() - datetime.timedelta(30,0,0)
```

In [6]:
```
cutoff
```

Out[6]:
```
Timestamp('2014-06-01 00:00:00')
```

In [7]:
```
churn_data['churn'] = (churn_data.last_trip_date < cutoff).astype(int)
```

In [8]:
```
churn_data.head()
```

Out[8]:

	avg_dist	avg_rating_by_driver	avg_rating_of_driver	avg_surge	city	last_trip_date	phone	signup_date	surge_pct	trips_in_first_30_days	luxury_car_u
0	3.67	5.0	4.7	1.10	King's Landing	2014-06-17	iPhone	2014-01-25	15.4	4	
1	8.26	5.0	5.0	1.00	Astapor	2014-05-05	Android	2014-01-29	0.0	0	F
2	0.77	5.0	4.3	1.00	Astapor	2014-01-07	iPhone	2014-01-06	0.0	3	F
3	2.36	4.9	4.6	1.14	King's Landing	2014-06-29	iPhone	2014-01-10	20.0	9	
4	3.13	4.9	4.4	1.19	Winterfell	2014-03-15	Android	2014-01-27	11.8	14	F

In [9]:
```
cat_cols = churn_data.select_dtypes('object').columns
```

In [10]:
```
churn_data[cat_cols].city.value_counts()
```

Out[10]:
```
Winterfell        23336
Astapor           16534
King's Landing    10130
Name: city, dtype: int64
```

In [11]:
```
churn_data[cat_cols].phone.value_counts()
```

Out[11]:
```
iPhone     34582
Android    15022
Name: phone, dtype: int64
```

In [12]:
```
cat_cols = list(cat_cols)
```

In [13]:
```
num_cols = list(churn_data.select_dtypes('float64').columns)
```

In [14]:
```
num_cols.append('trips_in_first_30_days')
```

In [15]:
```
num_cols
```

Out[15]:
```
['avg_dist',
 'avg_rating_by_driver',
 'avg_rating_of_driver',
 'avg_surge',
 'surge_pct',
 'weekday_pct',
 'trips_in_first_30_days']
```

In [16]:
```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
```

In [17]:
```
churn_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 13 columns):
avg_dist                50000 non-null float64
avg_rating_by_driver    49799 non-null float64
avg_rating_of_driver    41878 non-null float64
avg_surge               50000 non-null float64
city                    50000 non-null object
last_trip_date          50000 non-null datetime64[ns]
phone                   49604 non-null object
signup_date             50000 non-null datetime64[ns]
surge_pct               50000 non-null float64
trips_in_first_30_days  50000 non-null int64
luxury_car_user         50000 non-null bool
weekday_pct             50000 non-null float64
churn                   50000 non-null int32
dtypes: bool(1), datetime64[ns](2), float64(6), int32(1), int64(1), object(2)
memory usage: 4.4+ MB
```

In [18]:
```
pipeline_num = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaling',StandardScaler())
])
```

In [19]:
```
pipeline_cat = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
    ('encoding', OneHotEncoder(handle_unknown='ignore'))
])
```

In [20]:
```
preprocessor = ColumnTransformer(
    transformers=[
        ('num', pipeline_num, num_cols),
        ('cat', pipeline_cat, cat_cols)])
```

In [21]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
```

In [22]:
```
pipeline = Pipeline(steps=[('preprocessor',preprocessor),
                           ('classifier',RandomForestClassifier(n_estimators=10))])
```

In [23]:
```
trainX, testX, trainY, testY = train_test_split(churn_data, churn_data.churn)
```

In [24]:
```
pipeline.fit(trainX,trainY)
```

Out[24]:
```
Pipeline(memory=None,
     steps=[('preprocessor', ColumnTransformer(n_jobs=None, remainder='drop', sparse_threshold=0.3,
         transformer_weights=None,
         transformers=[('num', Pipeline(memory=None,
     steps=[('imputer', SimpleImputer(copy=True, fill_value=None, missing_values=nan,
       strategy='median', verbo...obs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False))])
```

In [25]:
```
pipeline.score(testX,testY)
```

Out[25]:
```
0.73968
```

In [30]:
```
param_grid = {
    'preprocessor__num__imputer__strategy': ['mean', 'median'],
    'classifier__n_estimators': [10,15,20]
    #'classifier__class_weight':["balanced",None]
}
```

In [31]:
```
from sklearn.model_selection import GridSearchCV
```

In [32]:
```
gs = GridSearchCV(pipeline, param_grid=param_grid, cv=5, n_jobs=-1)
```

In [33]:
```
gs.fit(trainX,trainY)
```

```
C:\Users\awant\Anaconda3\lib\site-packages\sklearn\externals\joblib\disk.py:122: UserWarning: Unable to delete folder C:\Users\awant\AppData\Local\Temp\joblib_memmapping_folder_20288_6514773715 after 5 tentatives.
  .format(folder_path, RM_SUBDIRS_N_RETRY))

---------------------------------------------------------------------------
PermissionError                           Traceback (most recent call last)
<ipython-input-33-551afae5d7d0> in <module>()
----> 1 gs.fit(trainX,trainY)

~\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py in fit(self, X, y, groups, **fit_params)
    720                 return results_container[0]
    721
--> 722             self._run_search(evaluate_candidates)
    723
    724         results = results_container[0]

~\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py in __exit__(self, exc_type, exc_value, traceback)
    730
    731     def __exit__(self, exc_type, exc_value, traceback):
--> 732         self._terminate_backend()
    733         self._managed_backend = False
    734

~\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py in _terminate_backend(self)
    760     def _terminate_backend(self):
    761         if self._backend is not None:
--> 762             self._backend.terminate()
    763
    764     def _dispatch(self, batch):

~\Anaconda3\lib\site-packages\sklearn\externals\joblib\_parallel_backends.py in terminate(self)
    524                 # in latter calls but we free as much memory as we can by deleting
    525                 # the shared memory
--> 526             delete_folder(self._workers._temp_folder)
    527             self._workers = None
    528

~\Anaconda3\lib\site-packages\sklearn\externals\joblib\disk.py in delete_folder(folder_path, onerror)
    113                 while True:
    114                     try:
--> 115                         shutil.rmtree(folder_path, False, None)
    116                         break
    117                     except (OSError, WindowsError):

~\Anaconda3\lib\shutil.py in rmtree(path, ignore_errors, onerror)
    492                 os.close(fd)
    493     else:
--> 494         return _rmtree_unsafe(path, onerror)
    495
    496 # Allow introspection of whether or not the hardening against symlink

~\Anaconda3\lib\shutil.py in _rmtree_unsafe(path, onerror)
    387                 os.unlink(fullname)
    388             except OSError:
--> 389                 onerror(os.unlink, fullname, sys.exc_info())
    390     try:
    391         os.rmdir(path)

~\Anaconda3\lib\shutil.py in _rmtree_unsafe(path, onerror)
    385         else:
    386             try:
--> 387                 os.unlink(fullname)
    388             except OSError:
    389                 onerror(os.unlink, fullname, sys.exc_info())

PermissionError: [WinError 32] The process cannot access the file because it is being used by another process: 'C:\\Users\\awant\\AppData\\Local\\Temp\\joblib_memmapping_folder_20288_6514773715\\20288-2169174037392-74dd2caacebf40ac8b8511bd1e3558a1.pkl'
```

In [ ]:
```
```