In [17]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
```

In [18]:

```
# Read the data
data_full = pd.read_csv('data/train.csv', index_col='Id')
```

In [19]:

```
print(data_full.shape)
data_full.head(3)
```

```
(1460, 80)
```

Out[19]:

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilitie
Id									
1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPu
2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPu
3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPu

3 rows × 80 columns

In [12]:

```
X_test_full = pd.read_csv('data/test.csv', index_col='Id')
print(X_test_full.shape)
X_test_full.head(3)
```

```
(1459, 79)
```

Out[12]:

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utili
Id									
1461	20	RH	80.0	11622	Pave	NaN	Reg	Lvl	All
1462	20	RL	81.0	14267	Pave	NaN	IR1	Lvl	All
1463	60	RL	74.0	13830	Pave	NaN	IR1	Lvl	All

3 rows × 79 columns

In [ ]:

In [15]:

```
# Remove rows with missing target,
data_full.dropna(axis=0, subset=['SalePrice'], inplace=True)
```

In [21]:

```
#separate target from predictors
y = data_full.SalePrice
X_full = data_full.drop(['SalePrice'], axis=1)
X_full.shape
```

Out[21]:

```
(1460, 79)
```

In [23]:

```
# Break off validation set from training data
X_train_full, X_valid_full, y_train, y_valid = train_test_split(X_full, y,
                                                                 train_size=0.8
, test_size=0.2,
                                                                 random_state=0
)
```

In [47]:

```
## object_column
s = (X_train_full.dtypes == 'object')
obj_cols = s[s].index
print(obj_cols.shape)
```

```
(43,)
```

In [48]:

```
# columns for categorical conversion
# "Cardinality" means the number of unique values in a column
# Select categorical columns with relatively low cardinality (convenient but a
rbitrary)
categorical_cols =  [colname for colname in X_train_full.columns
                     if X_train_full[colname].dtype == 'object' and
                     (X_train_full[colname].nunique() < 10)]

print(len(categorical_cols))
```

```
40
```

In [50]:

```
print(set(obj_cols)-set(categorical_cols))
```

```
{'Exterior1st', 'Neighborhood', 'Exterior2nd'}
```

In [89]:

```
# Select numerical columns
numerical_cols =  [colname for colname in X_train_full.columns if X_train_full
[colname].dtype != 'object']
print(len(numerical_col))
```

```
36
```

In [54]:

```
# Keep selected columns only
```

In [90]:

```
my_cols = numerical_cols + categorical_cols
print(len(my_col))
```

```
76
```

In [75]:

```
X_train = X_train_full[my_cols].copy()
X_valid = X_valid_full[my_cols].copy()
X_test = X_test_full[my_cols].copy()
```

In [ ]:

```
```

In [77]:

```
for col in numerical_col:
    if X_train[col].isnull().sum()>0:
        print(col, X_train[col].isnull().sum())
```

```
LotFrontage 212
MasVnrArea 6
GarageYrBlt 58
```

In [81]:

```
X_train.isnull().sum()[X_train.isnull().sum()>0]
#X_train.isnull().sum()[X_train.isnull()]
```

Out[81]:

```
LotFrontage      212
MasVnrArea         6
GarageYrBlt       58
Alley           1097
MasVnrType         6
BsmtQual          28
BsmtCond          28
BsmtExposure      28
BsmtFinType1      28
BsmtFinType2      29
Electrical         1
FireplaceQu      551
GarageType        58
GarageFinish      58
GarageQual        58
GarageCond        58
PoolQC          1164
Fence            954
MiscFeature     1119
dtype: int64
```

In [101]:

```
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
```

In [109]:

```
# Preprocessing for numerical data,
#When strategy == “constant”, fill_value is used to replace all occurrences of
missing_values.
#If left to the default, fill_value will be 0
numerical_transformer = SimpleImputer(strategy='constant')
```

In [80]:

```
```

In [125]:

```
# Preprocessing for categorical data
#If “most_frequent”, then replace missing using the most frequent value along
 each column.
#Can be used with strings or numeric data.
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='constant')), # handle NAN
    ('onehot', OneHotEncoder(handle_unknown='ignore'))    # handle
])
```

In [126]:

```
# Bundle preprocessing for numerical and categorical data
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_cols),
        ('cat', categorical_transformer, categorical_cols)
    ])
```

In [127]:

```
# Define model
model = RandomForestRegressor(n_estimators=100, random_state=0)
```

In [128]:

```
# Bundle preprocessing and modeling code in a pipeline
clf = Pipeline(steps=[('preprocessor', preprocessor),
                      ('model', model)
                     ])
```

In [159]:

```
from sklearn.model_selection import cross_val_score

# Multiply by -1 since sklearn calculates *negative* MAE
scores = -1 * cross_val_score(clf, X_train, y_train,
                              cv=5,
                              scoring='neg_mean_absolute_error')

print("MAE scores:\n", scores)

print("Average MAE score (across experiments):")
print(scores.mean())
```

```
MAE scores:
 [16342.85850427 20346.00730769 17490.18982906 19209.74957082
 16103.84643777]
Average MAE score (across experiments):
17898.530329921865
```

In [131]:

```
# Preprocessing of training data, fit model
clf.fit(X_train, y_train)

# Preprocessing of validation data, get predictions
preds = clf.predict(X_valid)

print('MAE:', mean_absolute_error(y_valid, preds))
```

```
MAE: 17621.3197260274
```

In [ ]:

```

```

In [155]:

```python
preds_test = clf.predict(X_test) # Your code here
```

In [143]:

```python
output = pd.DataFrame({'Id': X_test.index,
                       'SalePrice': preds_test})
output.to_csv('data/submission.csv', index=False)
```

In [190]:

```python
output.head(6)
```

Out[190]:

	Id	SalePrice
0	1461	127168.41
1	1462	154869.75
2	1463	182907.65
3	1464	182636.32
4	1465	199933.00
5	1466	185284.12