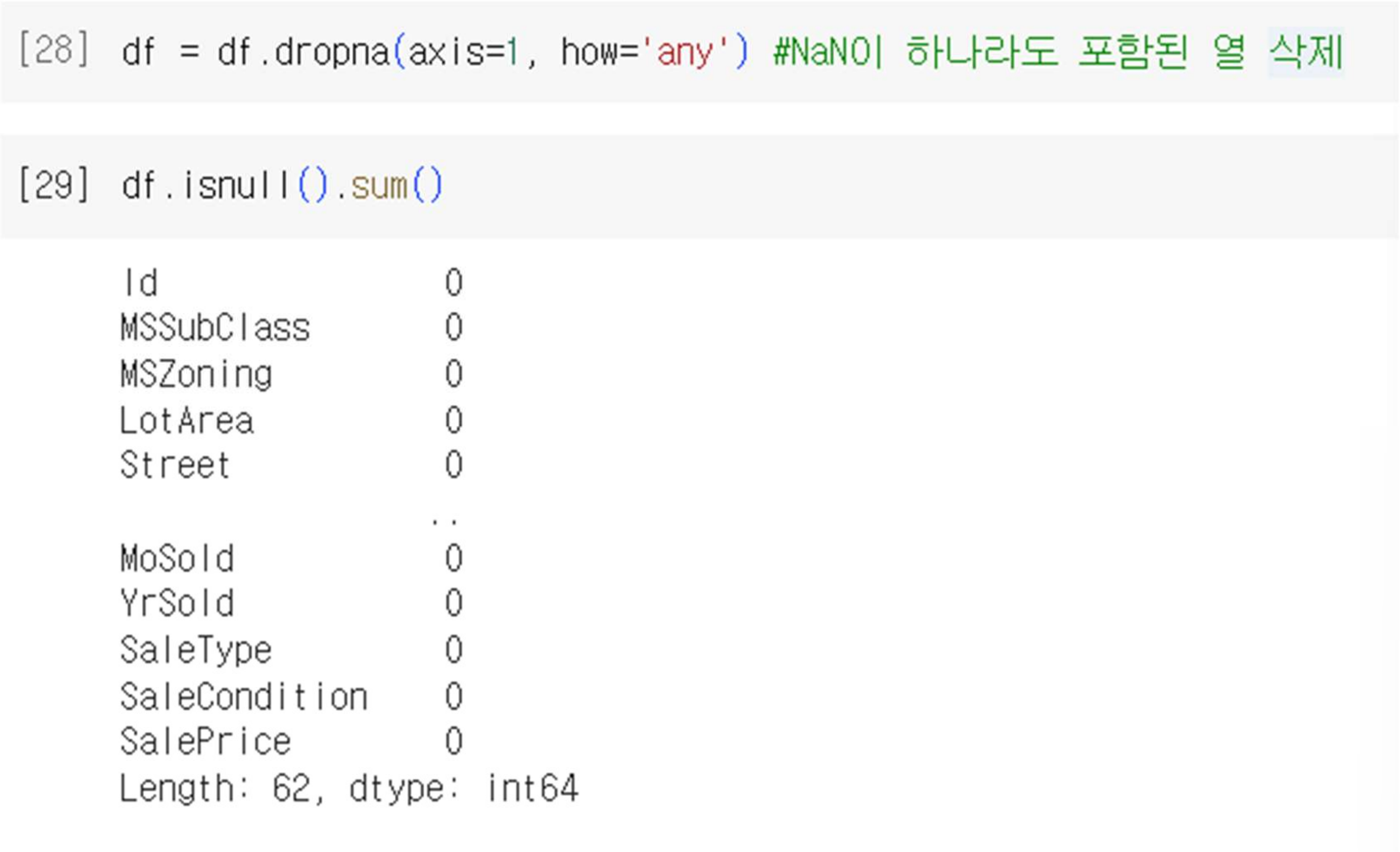
GDSC ML/DL Week06

서지현

Code Review

NaN -> df.dropna()로 삭제

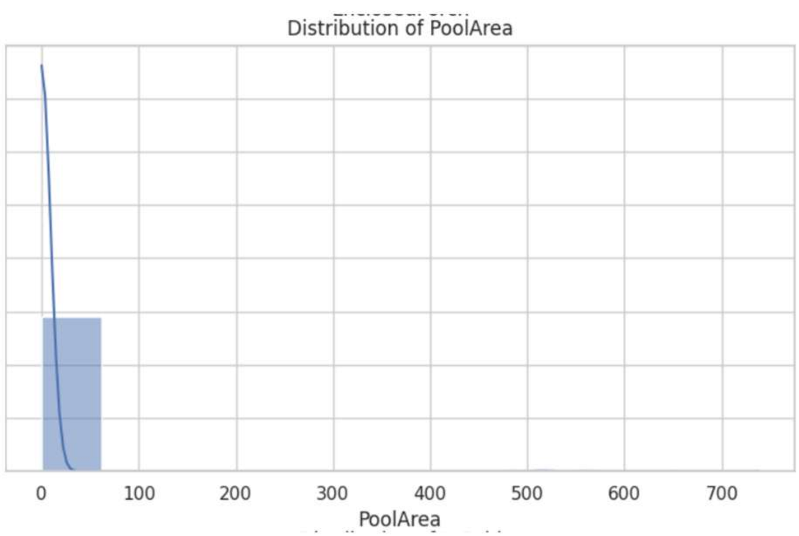


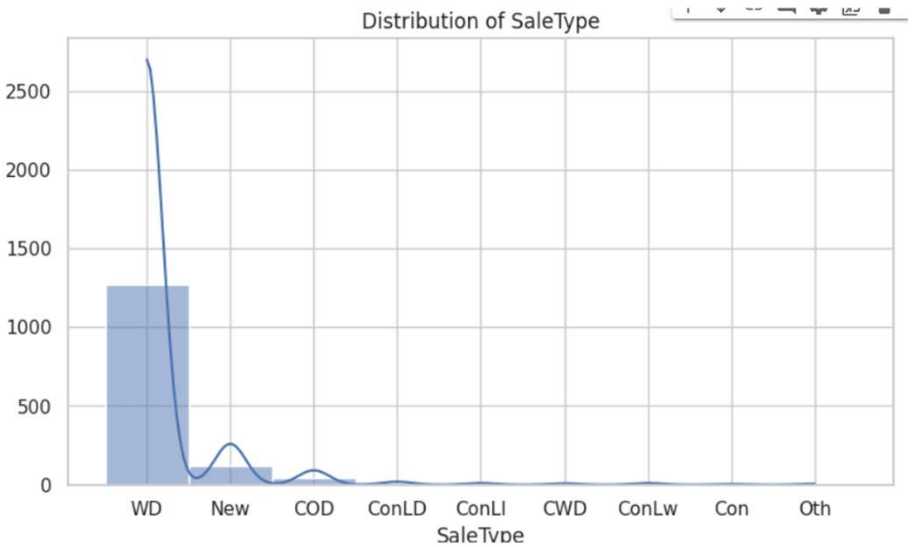
```
[28] df = df.dropna(axis=1, how='any') #NaN이 하나라도 포함된 열 삭제
```

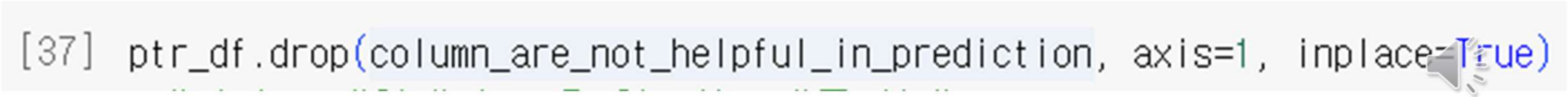
```
[29] df.isnull().sum()
```

```
Id               0
MSSubClass       0
MSZoning         0
LotArea          0
Street           0
                ..
MoSold           0
YrSold           0
SaleType         0
SaleCondition    0
SalePrice        0
Length: 62, dtype: int64
```

도움되지 않는 24개의 column 삭제

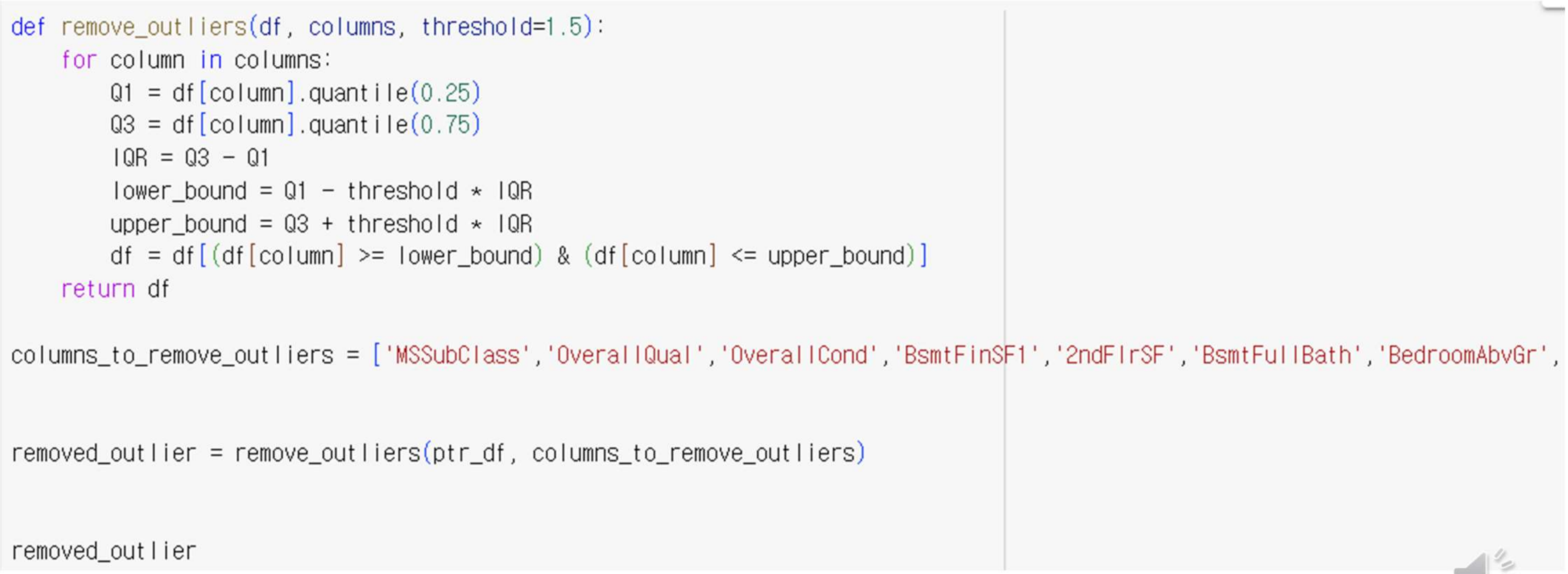






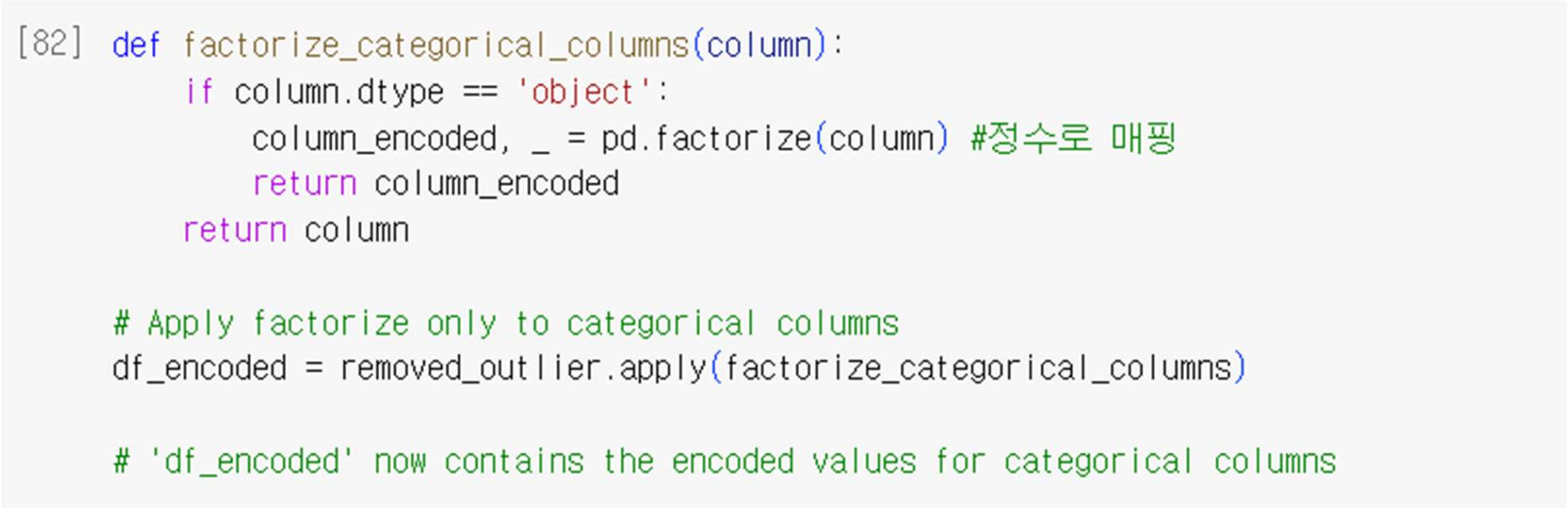
```
[37] ptr_df.drop(column_are_not_helpful_in_prediction, axis=1, inplace=True)
```

Outliers -> IQR 이용해서 분위 벗어나는 값 제거



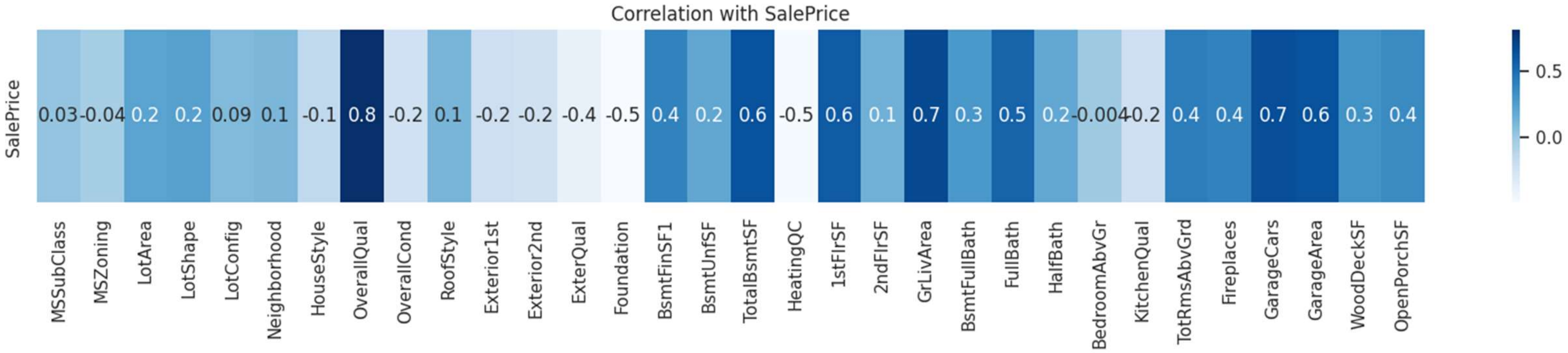
```
def remove_outliers(df, columns, threshold=1.5):
    for column in columns:
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - threshold * IQR
        upper_bound = Q3 + threshold * IQR
        df = df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]
    return df

columns_to_remove_outliers = ['MSSubClass','OverallQual','OverallCond','BsmtFinSF1','2ndFlrSF','BsmtFullBath','BedroomAbvGr',


removed_outlier = remove_outliers(ptr_df, columns_to_remove_outliers)


removed_outlier
```

범주형 -> pd.factorize()로 인코딩

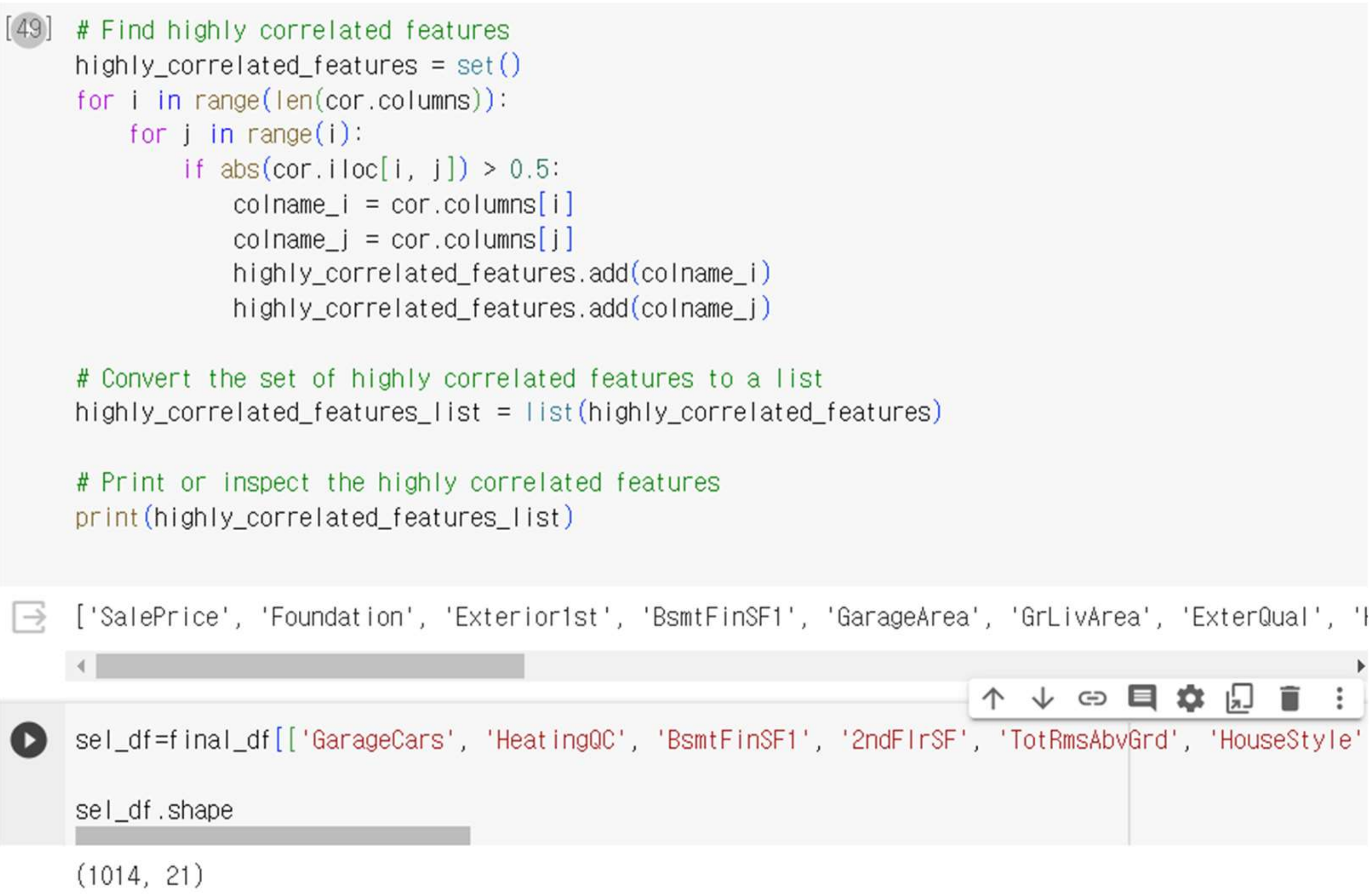


```
[82] def factorize_categorical_columns(column):
         if column.dtype == 'object':
             column_encoded, _ = pd.factorize(column) #정수로 매핑
             return column_encoded
         return column

     # Apply factorize only to categorical columns
     df_encoded = removed_outlier.apply(factorize_categorical_columns)

     # 'df_encoded' now contains the encoded values for categorical columns
```

히트맵-> SalePrice와 각 column간의 상관관계



상관계수가 높은 feature 추출



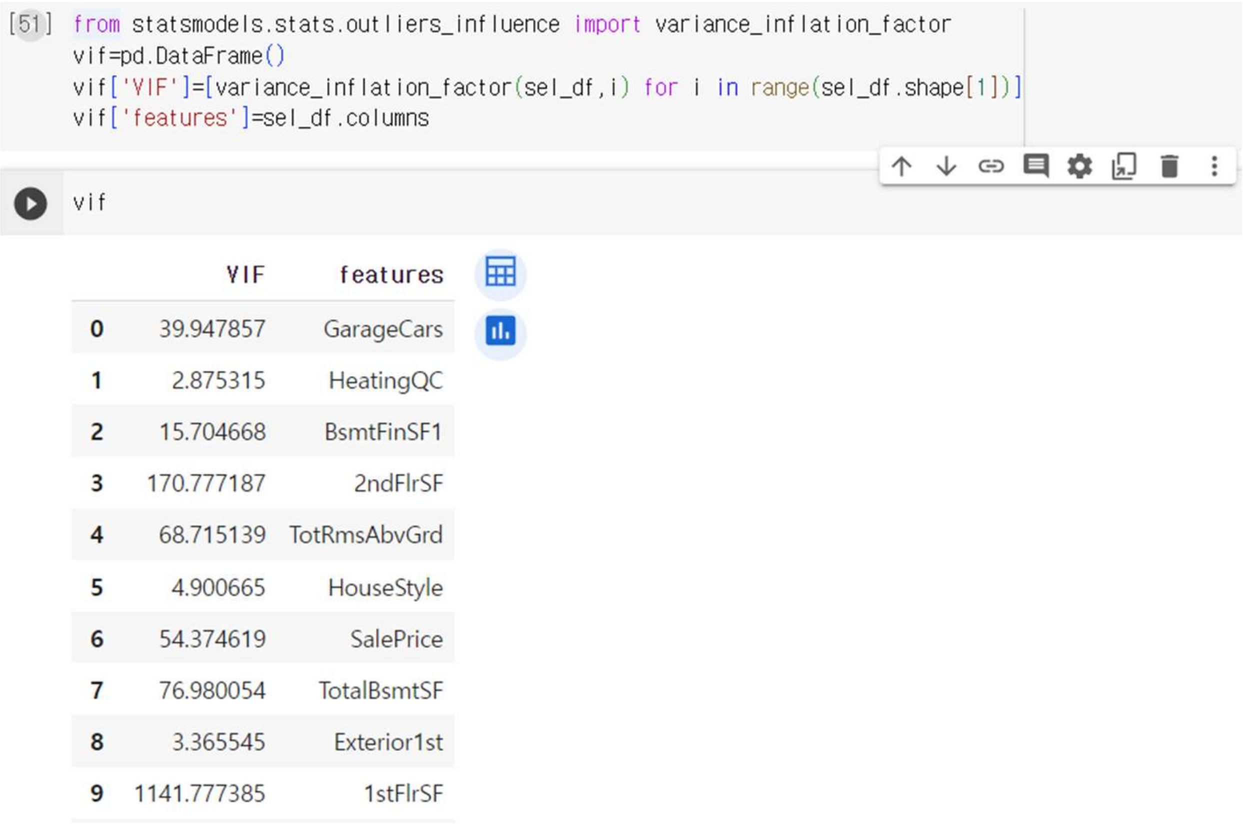
```python
# Find highly correlated features
highly_correlated_features = set()
for i in range(len(cor.columns)):
    for j in range(i):
        if abs(cor.iloc[i, j]) > 0.5:
            colname_i = cor.columns[i]
            colname_j = cor.columns[j]
            highly_correlated_features.add(colname_i)
            highly_correlated_features.add(colname_j)

# Convert the set of highly correlated features to a list
highly_correlated_features_list = list(highly_correlated_features)

# Print or inspect the highly correlated features
print(highly_correlated_features_list)
```

```
['SalePrice', 'Foundation', 'Exterior1st', 'BsmtFinSF1', 'GarageArea', 'GrLivArea', 'ExterQual', '
```

```python
sel_df=final_df[['GarageCars', 'HeatingQC', 'BsmtFinSF1', '2ndFlrSF', 'TotRmsAbvGrd', 'HouseStyle'

sel_df.shape
```

```
(1014, 21)
```

VIF 계산하여 multicollinearity 갖는 col 제거

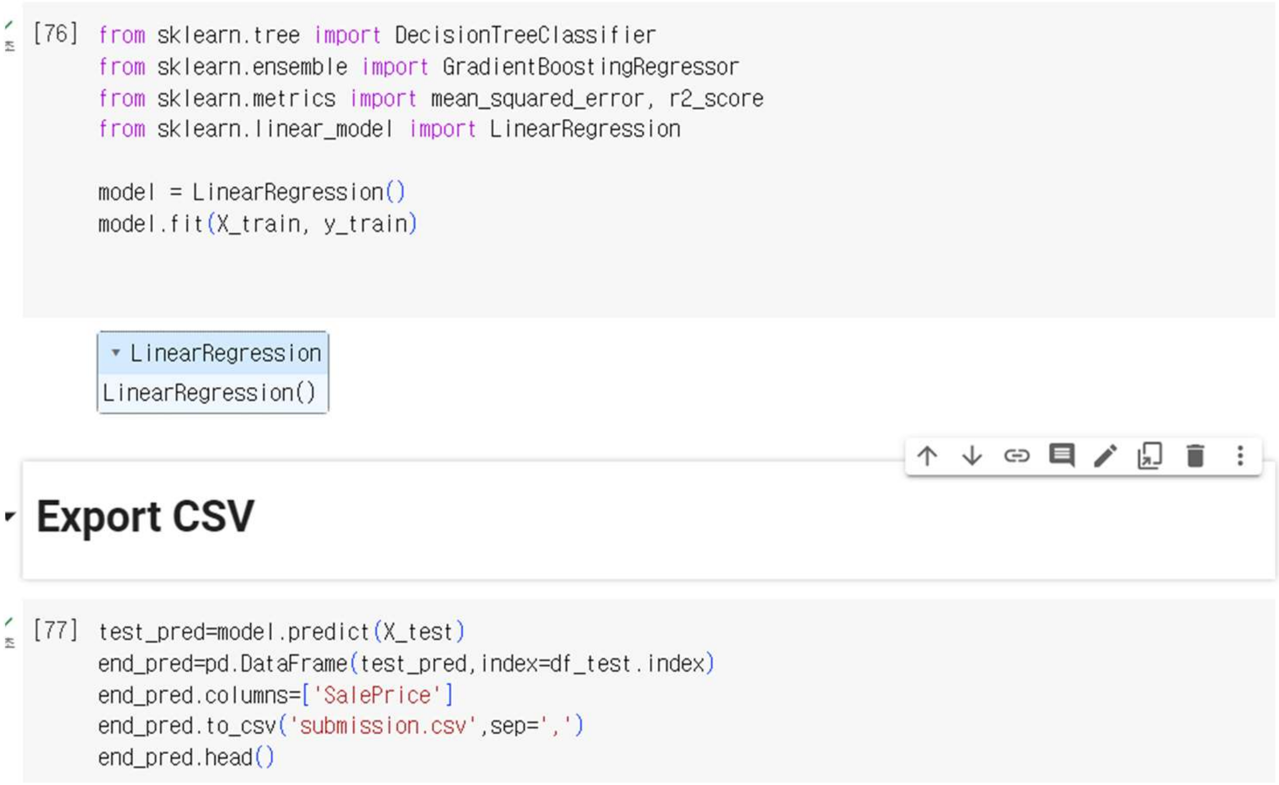


```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif=pd.DataFrame()
vif['VIF']=[variance_inflation_factor(sel_df,i) for i in range(sel_df.shape[1])]
vif['features']=sel_df.columns
```

```
vif
```

	VIF	features
0	39.947857	GarageCars
1	2.875315	HeatingQC
2	15.704668	BsmtFinSF1
3	170.777187	2ndFlrSF
4	68.715139	TotRmsAbvGrd
5	4.900665	HouseStyle
6	54.374619	SalePrice
7	76.980054	TotalBsmtSF
8	3.365545	Exterior1st
9	1141.777385	1stFlrSF

Linear Regression 진행



```
[76] from sklearn.tree import DecisionTreeClassifier
     from sklearn.ensemble import GradientBoostingRegressor
     from sklearn.metrics import mean_squared_error, r2_score
     from sklearn.linear_model import LinearRegression

     model = LinearRegression()
     model.fit(X_train, y_train)
```

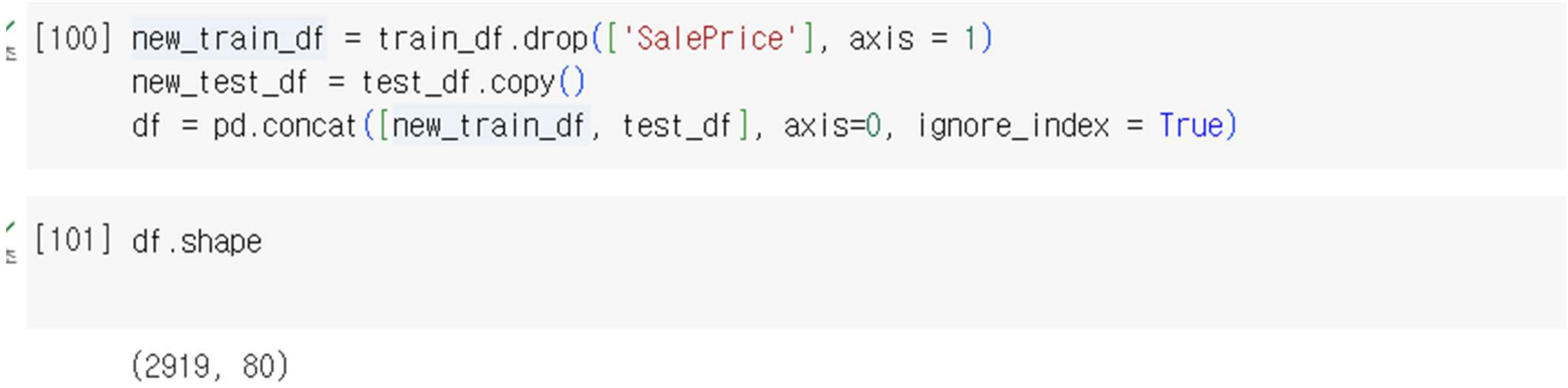
```
▾ LinearRegression
LinearRegression()
```

Export CSV

```
[77] test_pred=model.predict(X_test)
     end_pred=pd.DataFrame(test_pred,index=df_test.index)
     end_pred.columns=['SalePrice']
     end_pred.to_csv('submission.csv',sep=',')
     end_pred.head()
```

성능 개선

전처리 위해 train, test 데이터 프레임 합치기

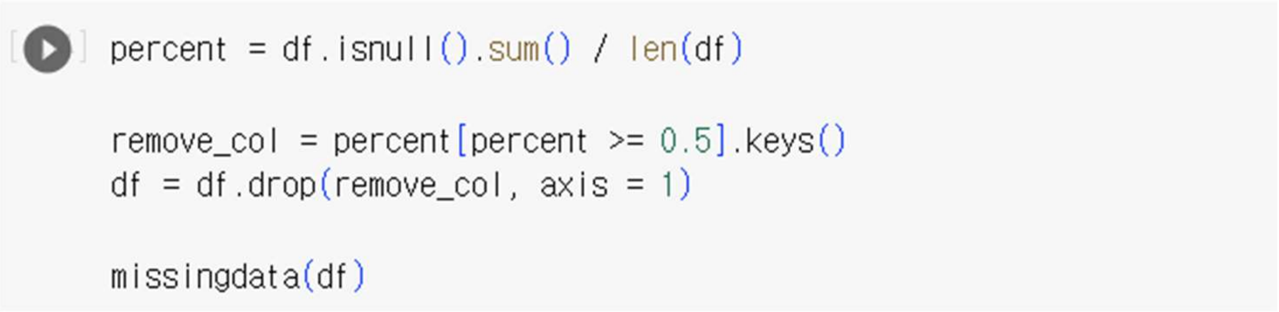


```
[100] new_train_df = train_df.drop(['SalePrice'], axis = 1)
      new_test_df = test_df.copy()
      df = pd.concat([new_train_df, test_df], axis=0, ignore_index = True)
```

```
[101] df.shape
```

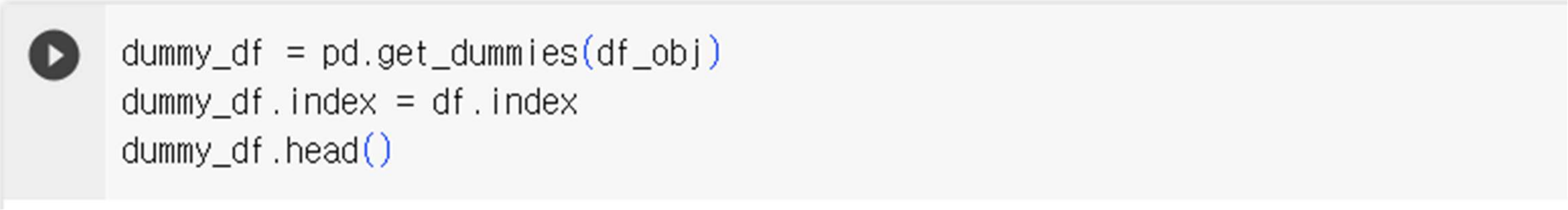
```
(2919, 80)
```

대부분이 NULL인 데이터 삭제



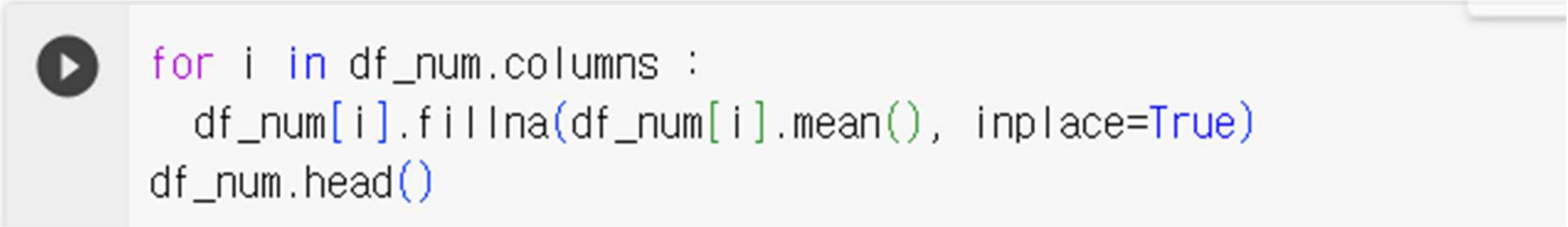
```
percent = df.isnull().sum() / len(df)

remove_col = percent[percent >= 0.5].keys()
df = df.drop(remove_col, axis = 1)

missingdata(df)
```

범주형 데이터 one-hot 인코딩



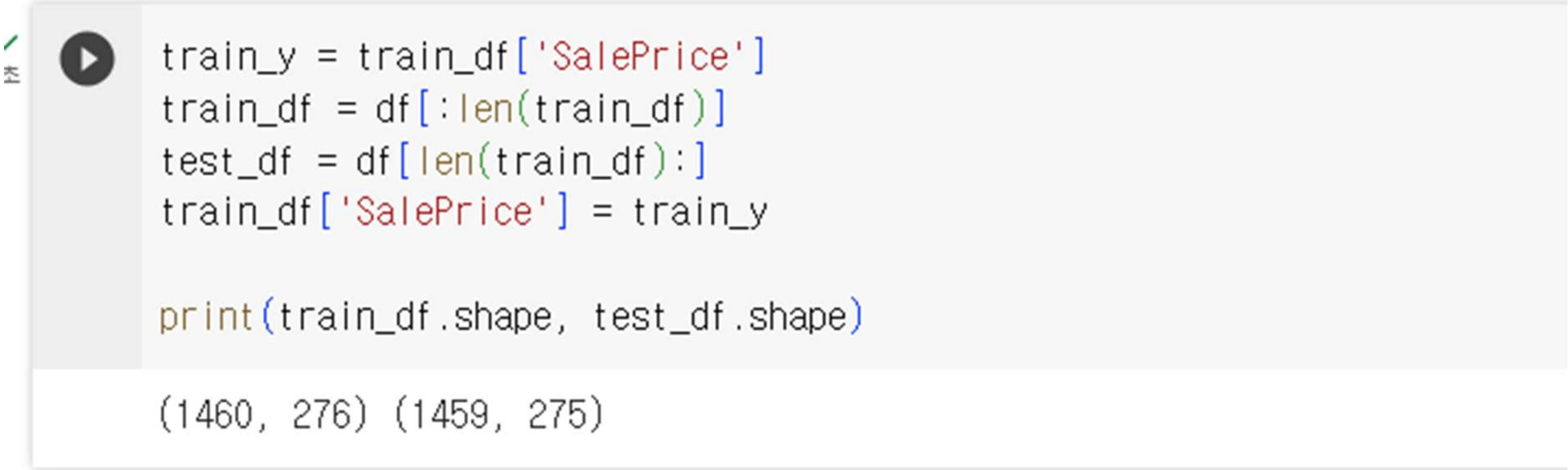
```
dummy_df = pd.get_dummies(df_obj)
dummy_df.index = df.index
dummy_df.head()
```

정수형 데이터의 null은 평균값으로 대체



```
for i in df_num.columns :
  df_num[i].fillna(df_num[i].mean(), inplace=True)
df_num.head()
```

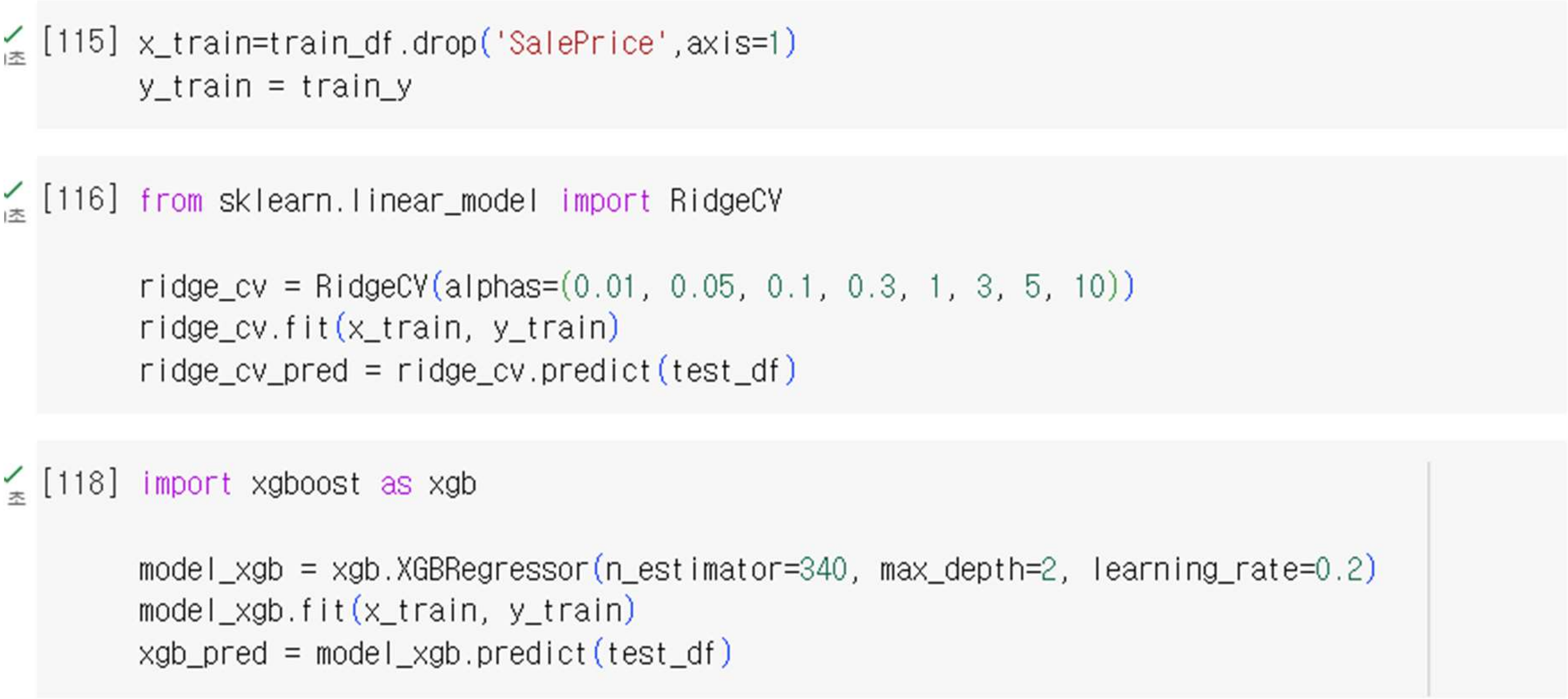
Train/Test 다시 분리



```
train_y = train_df['SalePrice']
train_df = df[:len(train_df)]
test_df = df[len(train_df):]
train_df['SalePrice'] = train_y

print(train_df.shape, test_df.shape)
```

```
(1460, 276) (1459, 275)
```

모델: RidgeCV와 XGBboost의 평균 사용

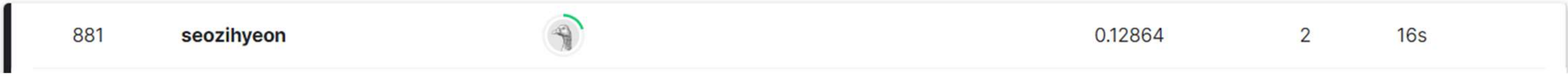


```
[115] x_train=train_df.drop('SalePrice',axis=1)
      y_train = train_y
```

```
[116] from sklearn.linear_model import RidgeCV

      ridge_cv = RidgeCV(alphas=(0.01, 0.05, 0.1, 0.3, 1, 3, 5, 10))
      ridge_cv.fit(x_train, y_train)
      ridge_cv_pred = ridge_cv.predict(test_df)
```

```
[118] import xgboost as xgb

      model_xgb = xgb.XGBRegressor(n_estimator=340, max_depth=2, learning_rate=0.2)
      model_xgb.fit(x_train, y_train)
      xgb_pred = model_xgb.predict(test_df)
```

결과: 0.12864



881	seozihyeon		0.12864	2	16s