predict

December 6, 2023

[ ]:
```
### Libraries

# Misc.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import colormaps as cm
import seaborn as sns
import datetime as dt
import os

# Preprocessing
from sklearn.preprocessing import OneHotEncoder, MinMaxScaler, StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.utils import class_weight

# Dimension Reduction
from sklearn.decomposition import PCA
from sklearn.manifold import Isomap, SpectralEmbedding, TSNE

# Models
from sklearn.neighbors import KNeighborsClassifier, NearestNeighbors
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier, RandomForestClassifier
import xgboost as xgb

# Evaluation
from sklearn.metrics import f1_score, accuracy_score
```

```
Intel MKL WARNING: Support of Intel(R) Streaming SIMD Extensions 4.2 (Intel(R)
SSE4.2) enabled only processors has been deprecated. Intel oneAPI Math Kernel
Library 2025.0 will require Intel(R) Advanced Vector Extensions (Intel(R) AVX)
instructions.
Intel MKL WARNING: Support of Intel(R) Streaming SIMD Extensions 4.2 (Intel(R)
SSE4.2) enabled only processors has been deprecated. Intel oneAPI Math Kernel
Library 2025.0 will require Intel(R) Advanced Vector Extensions (Intel(R) AVX)
instructions.
```

[ ]:
```
### Read data

data_path = f'{os.path.dirname(os.getcwd())}/data'
train_df = pd.read_csv(f'{data_path}/train.csv')
test_df = pd.read_csv(f'{data_path}/test.csv')
```

[ ]:
```
### Clean Information

def clean(df: pd.DataFrame):
    # Drop NaNs from beds and bathrooms_text columns
    df.dropna(subset = ['beds', 'bathrooms_text'], inplace = True)

    # Group hotel and shared rooms into 'other' category
    rooms_regrouped = df['room_type'].where((df['room_type'] == 'Entire home/
 ↪apt') | (df['room_type'] == 'Private room'), 'Other')
    df['rooms_regrouped'] = rooms_regrouped
    df['entire_bin'] = np.where(df['rooms_regrouped'] == 'Entire home/apt', 1,␣
 ↪0)
    df['private_bin'] = np.where(df['rooms_regrouped'] == 'Private room', 1, 0)
    df['other_room_bin'] = np.where(df['rooms_regrouped'] == 'Other', 1, 0)

    # Extract 'shared' keyword from bathrooms_text column
    def shared_bathrooms(row):
        if type(row['bathrooms_text']) is not str or 'shared' not in␣
 ↪row['bathrooms_text']:
            return 0
        return 1
    df['bathrooms_shared'] = df.apply(shared_bathrooms, axis = 1)

    # Extract number of baths from bathrooms_text column
    def extract_num(row):
        char_arr = np.array(row['bathrooms_text'].split())
        res = char_arr[np.char.isnumeric(char_arr)].astype(float)
        return res[0] if res.size != 0 else 1 # HOW TO IMPUTE TEXT-ONLY SAMPLES
    df['bathrooms_num'] = df.apply(extract_num, axis = 1)

    # Extract number of amenities from amenities column
    def extract_amenities(row):
        return set(row['amenities'][2:-2].split('''", "'''))
    df['amenities_ref'] = df.apply(extract_amenities, axis = 1)
    def count_amenities(row):
        return len(row['amenities_ref'])
    df['amenities_count'] = df.apply(count_amenities, axis = 1)

clean(train_df)
clean(test_df)
```

[ ]:
```
### Pull out relevant features identified in EDA

features = ['host_listings_count',␣
 ↪'calculated_host_listings_count_private_rooms', 'entire_bin', 'private_bin',␣
 ↪'other_room_bin', 'accommodates', 'bathrooms_shared', 'bathrooms_num',
            'beds', 'amenities_count', 'latitude', 'longitude',␣
 ↪'number_of_reviews', 'number_of_reviews_ltm', 'number_of_reviews_l30d',␣
 ↪'availability_30', 'availability_365',
            'neighbourhood_cleansed', 'property_type', 'price']
target = 'price'

### Split training data
X_train, X_test, y_train = train_df[features], test_df[features[:-1]],␣
 ↪train_df[target]
```

[ ]:
```
impute_features = ['host_listings_count',␣
 ↪'calculated_host_listings_count_private_rooms', 'entire_bin', 'private_bin',␣
 ↪'other_room_bin', 'accommodates', 'bathrooms_shared',
                   'bathrooms_num', 'beds', 'amenities_count', 'latitude',␣
 ↪'longitude', 'number_of_reviews', 'number_of_reviews_ltm',␣
 ↪'number_of_reviews_l30d', 'availability_30',
                   'availability_365']
regional_metrics_knn_df = X_train[impute_features]
regional_metrics_knn_df = MinMaxScaler().fit_transform(X =␣
 ↪regional_metrics_knn_df)
nearest_neighbors = NearestNeighbors(n_neighbors = 16).fit(X =␣
 ↪regional_metrics_knn_df)

def impute_nans(row, feature, metric):
    if np.isnan(row[feature]):
        neighbors = nearest_neighbors.kneighbors(row[impute_features].values.
 ↪reshape(1, -1), return_distance = False)
        if metric == 'mean':
            return X_train[feature].iloc[list(neighbors[0])].mean()
        if metric == 'median':
            return X_train[feature].iloc[list(neighbors[0])].median()
    else:
        return row[feature]

threshold = 20

mean_neighborhood_price = X_train[['neighbourhood_cleansed', 'price']].
 ↪groupby(by = 'neighbourhood_cleansed').mean().to_dict()['price']
median_neighborhood_price = X_train[['neighbourhood_cleansed', 'price']].
 ↪groupby(by = 'neighbourhood_cleansed').median().to_dict()['price']
neighborhood_counts = X_train['neighbourhood_cleansed'].value_counts()
```

```
for neighborhood in neighborhood_counts.index:
    if neighborhood_counts[neighborhood] < threshold:
        mean_neighborhood_price.pop(neighborhood, None)
        median_neighborhood_price.pop(neighborhood, None)
X_train['mean_neighborhood_price'] = X_train['neighbourhood_cleansed'].
 ↪map(mean_neighborhood_price)
X_train['median_neighborhood_price'] = X_train['neighbourhood_cleansed'].
 ↪map(median_neighborhood_price)
X_train['mean_neighborhood_price'] = X_train.apply(impute_nans, args =␣
 ↪('mean_neighborhood_price', 'mean'), axis = 1)
X_train['median_neighborhood_price'] = X_train.apply(impute_nans, args =␣
 ↪('median_neighborhood_price', 'median'), axis = 1)

threshold = 20

mean_property_type_price = X_train[['property_type', 'price']].groupby(by =␣
 ↪'property_type').mean().to_dict()['price']
median_property_type_price = X_train[['property_type', 'price']].groupby(by =␣
 ↪'property_type').median().to_dict()['price']
property_counts = X_train['property_type'].value_counts()
for property in property_counts.index:
    if property_counts[property] < threshold:
        mean_property_type_price.pop(property, None)
        median_property_type_price.pop(property, None)
X_train['mean_property_type_price'] = X_train['property_type'].
 ↪map(mean_property_type_price)
X_train['median_property_type_price'] = X_train['property_type'].
 ↪map(median_property_type_price)
X_train['mean_property_type_price'] = X_train.apply(impute_nans, args =␣
 ↪('mean_property_type_price', 'mean'), axis = 1)
X_train['median_property_type_price'] = X_train.apply(impute_nans, args =␣
 ↪('median_property_type_price', 'median'), axis = 1)
```

```
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:767: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:605: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:614: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
pd.SparseDtype)` instead.
```

```
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:767: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:605: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:614: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/var/folders/_h/l_5070nj7zncym2g5tmhyvs40000gn/T/ipykernel_74615/820710769.py:27
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_train['mean_neighborhood_price'] =
X_train['neighbourhood_cleansed'].map(mean_neighborhood_price)
/var/folders/_h/l_5070nj7zncym2g5tmhyvs40000gn/T/ipykernel_74615/820710769.py:28
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_train['median_neighborhood_price'] =
X_train['neighbourhood_cleansed'].map(median_neighborhood_price)
/var/folders/_h/l_5070nj7zncym2g5tmhyvs40000gn/T/ipykernel_74615/820710769.py:29
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_train['mean_neighborhood_price'] = X_train.apply(impute_nans, args =
('mean_neighborhood_price', 'mean'), axis = 1)
/var/folders/_h/l_5070nj7zncym2g5tmhyvs40000gn/T/ipykernel_74615/820710769.py:30
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_train['median_neighborhood_price'] = X_train.apply(impute_nans, args =
('median_neighborhood_price', 'median'), axis = 1)
/var/folders/_h/l_5070nj7zncym2g5tmhyvs40000gn/T/ipykernel_74615/820710769.py:41
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_train['mean_property_type_price'] =
X_train['property_type'].map(mean_property_type_price)
/var/folders/_h/l_5070nj7zncym2g5tmhyvs40000gn/T/ipykernel_74615/820710769.py:42
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_train['median_property_type_price'] =
X_train['property_type'].map(median_property_type_price)
/var/folders/_h/l_5070nj7zncym2g5tmhyvs40000gn/T/ipykernel_74615/820710769.py:43
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_train['mean_property_type_price'] = X_train.apply(impute_nans, args =
('mean_property_type_price', 'mean'), axis = 1)
/var/folders/_h/l_5070nj7zncym2g5tmhyvs40000gn/T/ipykernel_74615/820710769.py:44
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_train['median_property_type_price'] = X_train.apply(impute_nans, args =
('median_property_type_price', 'median'), axis = 1)
```

[ ]:
```
### Extraction of mean/median prices by categorical features

nearest_neighbors = NearestNeighbors(n_neighbors = 16).fit(X =␣
  ↪regional_metrics_knn_df)
```

```
X_test['mean_neighborhood_price'] = X_test['neighbourhood_cleansed'].
 ↪map(mean_neighborhood_price)
X_test['mean_neighborhood_price'] = X_test.apply(impute_nans, args =␣
 ↪('mean_neighborhood_price', 'mean'), axis = 1)

X_test['median_neighborhood_price'] = X_test['neighbourhood_cleansed'].
 ↪map(median_neighborhood_price)
X_test['median_neighborhood_price'] = X_test.apply(impute_nans, args =␣
 ↪('median_neighborhood_price', 'median'), axis = 1)

X_test['mean_property_type_price'] = X_test['property_type'].
 ↪map(mean_property_type_price)
X_test['mean_property_type_price'] = X_test.apply(impute_nans, args =␣
 ↪('mean_property_type_price', 'mean'), axis = 1)

X_test['median_property_type_price'] = X_test['property_type'].
 ↪map(median_property_type_price)
X_test['median_property_type_price'] = X_test.apply(impute_nans, args =␣
 ↪('median_property_type_price', 'median'), axis = 1)

#X_test['mean_host_id_price'] = X_test['host_id'].map(mean_host_id_price)
#X_test['mean_host_id_price'] = X_test.apply(impute_nans, args =␣
 ↪('mean_host_id_price', 'mean'), axis = 1)

#X_test['median_host_id_price'] = X_test['host_id'].map(median_host_id_price)
#X_test['median_host_id_price'] = X_test.apply(impute_nans, args =␣
 ↪('median_host_id_price', 'mean'), axis = 1)

X_train = X_train.drop(['neighbourhood_cleansed', 'property_type', 'price'],␣
 ↪axis = 1)
X_test = X_test.drop(['neighbourhood_cleansed', 'property_type'], axis = 1)

ss = StandardScaler().fit(X = X_train)
X_train = pd.DataFrame(ss.transform(X = X_train), index = X_train.index,␣
 ↪columns = X_train.columns)
X_test = pd.DataFrame(ss.transform(X = X_test), index = X_test.index, columns =␣
 ↪X_test.columns)
```

```
/var/folders/_h/l_5070nj7zncym2g5tmhyvs40000gn/T/ipykernel_74615/1246814453.py:5
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_test['mean_neighborhood_price'] =
X_test['neighbourhood_cleansed'].map(mean_neighborhood_price)
```

```
/var/folders/_h/l_5070nj7zncym2g5tmhyvs40000gn/T/ipykernel_74615/1246814453.py:6
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_test['mean_neighborhood_price'] = X_test.apply(impute_nans, args =
('mean_neighborhood_price', 'mean'), axis = 1)
/var/folders/_h/l_5070nj7zncym2g5tmhyvs40000gn/T/ipykernel_74615/1246814453.py:8
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_test['median_neighborhood_price'] =
X_test['neighbourhood_cleansed'].map(median_neighborhood_price)
/var/folders/_h/l_5070nj7zncym2g5tmhyvs40000gn/T/ipykernel_74615/1246814453.py:9
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_test['median_neighborhood_price'] = X_test.apply(impute_nans, args =
('median_neighborhood_price', 'median'), axis = 1)
/var/folders/_h/l_5070nj7zncym2g5tmhyvs40000gn/T/ipykernel_74615/1246814453.py:1
1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_test['mean_property_type_price'] =
X_test['property_type'].map(mean_property_type_price)
/var/folders/_h/l_5070nj7zncym2g5tmhyvs40000gn/T/ipykernel_74615/1246814453.py:1
2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_test['mean_property_type_price'] = X_test.apply(impute_nans, args =
('mean_property_type_price', 'mean'), axis = 1)
/var/folders/_h/l_5070nj7zncym2g5tmhyvs40000gn/T/ipykernel_74615/1246814453.py:1
4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_test['median_property_type_price'] =
X_test['property_type'].map(median_property_type_price)
/var/folders/_h/l_5070nj7zncym2g5tmhyvs40000gn/T/ipykernel_74615/1246814453.py:1
5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_test['median_property_type_price'] = X_test.apply(impute_nans, args =
('median_property_type_price', 'median'), axis = 1)
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:767: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:605: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:614: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:767: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:605: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:614: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:767: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
```

```
pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:605: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:614: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
```

[ ]:
```
#from sklearn.decomposition import PCA
#from sklearn.manifold import Isomap, SpectralEmbedding, TSNE
restricted_size_features_geo = ['accommodates', 'bathrooms_num', 'beds',␣
 ↪'latitude', 'longitude']
combined = pd.concat([X_train[restricted_size_features_geo],␣
 ↪X_test[restricted_size_features_geo]])

# Spectral (knn = 15)
seknn = SpectralEmbedding(n_components = 2, affinity = 'nearest_neighbors',␣
 ↪n_neighbors = 250).fit_transform(X = combined)

# Isomap
iso = Isomap(n_components = 2, n_neighbors = 250).fit_transform(X = combined)

# TSNE
tsne = TSNE(n_components = 2, perplexity = 50).fit_transform(X = combined)
```

```
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:767: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:605: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:614: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:767: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
```

```
pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:605: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:614: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:767: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:605: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:614: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
```

[ ]:
```
pca = PCA(n_components = 2).fit_transform(X = combined)
```

```
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:767: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:605: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/sklearn/utils/validation.py:614: FutureWarning: is_sparse is deprecated
and will be removed in a future version. Check `isinstance(dtype,
pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
```

[ ]:
```
X_train['spec_x'] = seknn[:,0][:15314]
X_train['spec_y'] = seknn[:,1][:15314]
```

```
X_train['iso_x'] = iso[:,0][:15314]
X_train['iso_y'] = iso[:,1][:15314]
X_train['tsne_x'] = tsne[:,0][:15314]
X_train['tsne_y'] = tsne[:,1][:15314]
X_train['pca_x'] = pca[:,0][:15314]
X_train['pca_y'] = pca[:,1][:15314]

X_test['spec_x'] = seknn[:,0][15314:]
X_test['spec_y'] = seknn[:,1][15314:]
X_test['iso_x'] = iso[:,0][15314:]
X_test['iso_y'] = iso[:,1][15314:]
X_test['tsne_x'] = tsne[:,0][15314:]
X_test['tsne_y'] = tsne[:,1][15314:]
X_test['pca_x'] = pca[:,0][15314:]
X_test['pca_y'] = pca[:,1][15314:]

#ss = StandardScaler().fit(X = X_train)
#X_train = pd.DataFrame(ss.transform(X = X_train), index = X_train.index,␣
 ↪columns = X_train.columns)
#X_test = pd.DataFrame(ss.transform(X = X_test), index = X_test.index, columns␣
 ↪= X_test.columns)
```

[ ]:
```
classes_weights = class_weight.compute_sample_weight(class_weight = 'balanced',␣
 ↪y = y_train)
xgb_clf = xgb.XGBClassifier(n_estimators = 500, max_depth = 8, learning_rate =␣
 ↪0.1, reg_lambda = 0.7, objective = 'binary:logistic').fit(X = X_train, y =␣
 ↪y_train, sample_weight = classes_weights)
y_pred_xgb = xgb_clf.predict(X = X_test)
pred = pd.DataFrame(y_pred_xgb, index = X_test.index, columns = ['price'])
pred.index.name = 'id'
pred.to_csv('submission.csv')
```

```
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/xgboost/data.py:299: FutureWarning: is_sparse is deprecated and will be
removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(dtype):
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/xgboost/data.py:301: FutureWarning: is_categorical_dtype is deprecated
and will be removed in a future version. Use isinstance(dtype, CategoricalDtype)
instead
  elif is_categorical_dtype(dtype) and enable_categorical:
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/xgboost/data.py:332: FutureWarning: is_categorical_dtype is deprecated
and will be removed in a future version. Use isinstance(dtype, CategoricalDtype)
instead
  if is_categorical_dtype(dtype)
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
```

```
packages/xgboost/data.py:323: FutureWarning: is_categorical_dtype is deprecated
and will be removed in a future version. Use isinstance(dtype, CategoricalDtype)
instead
  return is_int or is_bool or is_float or is_categorical_dtype(dtype)
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/xgboost/data.py:427: FutureWarning: is_sparse is deprecated and will be
removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(data):
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/xgboost/data.py:299: FutureWarning: is_sparse is deprecated and will be
removed in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(dtype):
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/xgboost/data.py:301: FutureWarning: is_categorical_dtype is deprecated
and will be removed in a future version. Use isinstance(dtype, CategoricalDtype)
instead
  elif is_categorical_dtype(dtype) and enable_categorical:
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/xgboost/data.py:332: FutureWarning: is_categorical_dtype is deprecated
and will be removed in a future version. Use isinstance(dtype, CategoricalDtype)
instead
  if is_categorical_dtype(dtype)
/opt/anaconda3/envs/cs671_final_project/lib/python3.9/site-
packages/xgboost/data.py:323: FutureWarning: is_categorical_dtype is deprecated
and will be removed in a future version. Use isinstance(dtype, CategoricalDtype)
instead
  return is_int or is_bool or is_float or is_categorical_dtype(dtype)
```

rf_clf = RandomForestClassifier(n_estimators = 500).fit(X = X_train, y = y_train) y_pred_rf = rf_clf.predict(X = X_test) pred = pd.DataFrame(y_pred_rf, index = X_test.index, columns = ['price']) pred.index.name = 'id' pred.to_csv('submission.csv')

[ ]:
```
pred.value_counts()
```

[ ]:
```
price
0        1669
3        1212
1        1208
2        1030
4         734
5         438
Name: count, dtype: int64
```

[ ]: