```
import os
import tensorflow as tf
import pandas as pd
from tensorflow.keras import utils
from tensorflow.keras import datasets, layers, models
import seaborn as sns

import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import convolve2d

tf.compat.v1.logging.set_verbosity(tf.compat.v1.logging.ERROR)

from google.colab import drive
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor
from sklearn.ensemble import BaggingRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.svm import SVR
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive",
force_remount=True).
```

```
# set the path to the folder containing the CSV files
folder_path = "/content/drive/MyDrive/Datasets"

# get a list of all CSV files in the folder
csv_files = [file for file in os.listdir(folder_path) if file.endswith('.csv')]

combined_data = pd.DataFrame()
```

```
for file in csv_files:
    file_path = os.path.join(folder_path, file)
    data = pd.read_csv(file_path)
    combined_data = combined_data.append(data, ignore_index=True)

re_str = 'photos/'
house_data = combined_data.drop(combined_data.columns[
    combined_data.columns.str.contains(re_str)], axis=1)
house_data = house_data.drop(combined_data.columns[
    combined_data.columns.str.contains("address/community")], axis=1)
```

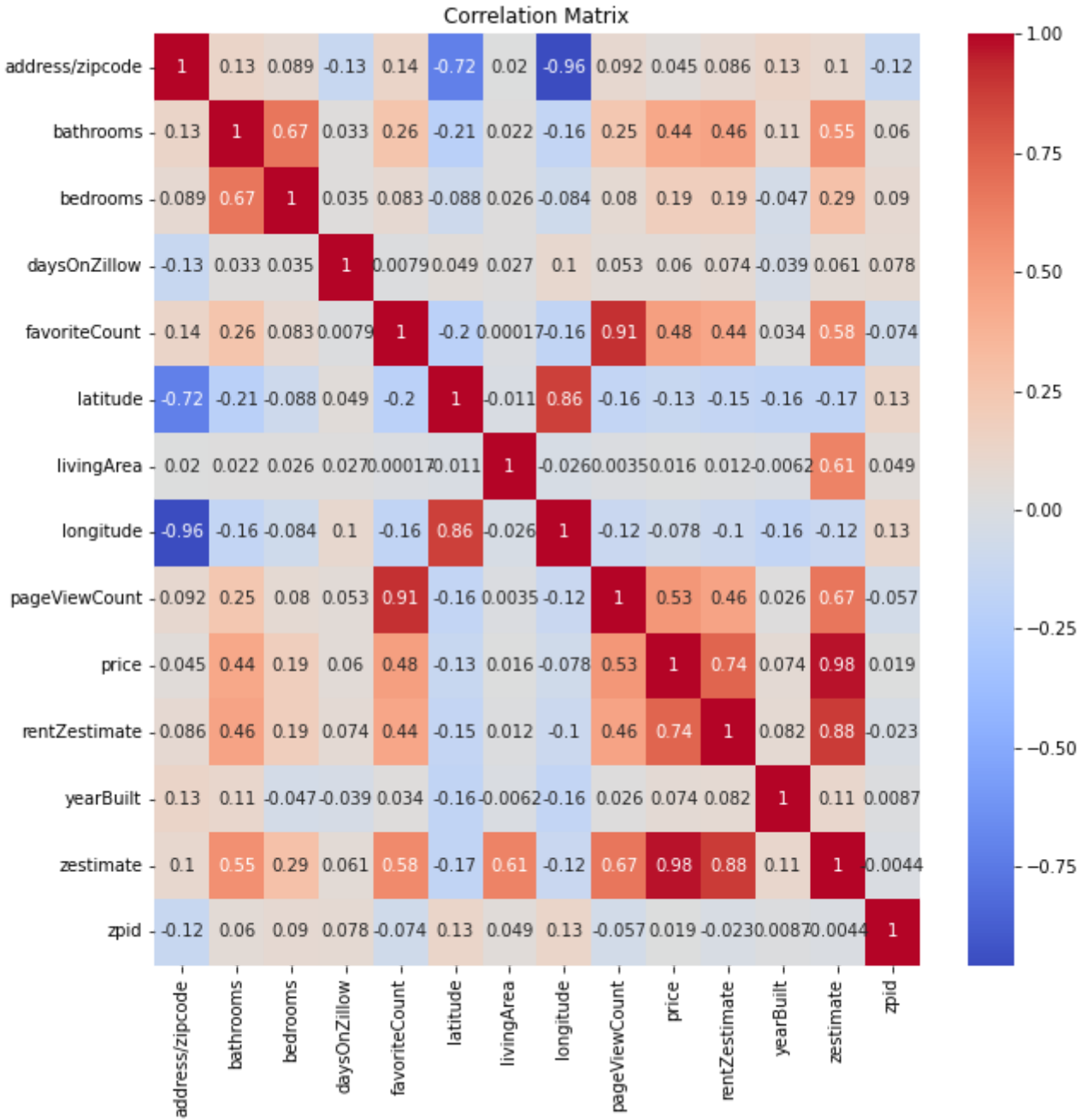
```
house_data['bathrooms'].fillna(combined_data['bathrooms'].mean(), inplace=True)
house_data['bedrooms'].fillna(combined_data['bedrooms'].mean(), inplace=True)
```

```
X = house_data[['address/zipcode','bathrooms', 'bedrooms']]
y = house_data['price']
```

```
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=25)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2,
                                                  random_state=25)
```

```
corr_matrix = house_data.corr()

plt.figure(figsize=(10, 10))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```




```
pip install lazypredict
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: lazypredict in /usr/local/lib/python3.9/dist-packages (0.2.12)
Requirement already satisfied: tqdm in /usr/local/lib/python3.9/dist-packages (from lazypredict) (4.65.0)
Requirement already satisfied: xgboost in /usr/local/lib/python3.9/dist-packages (from lazypredict) (1.7.4)
Requirement already satisfied: click in /usr/local/lib/python3.9/dist-packages (from lazypredict) (8.1.3)
Requirement already satisfied: pandas in /usr/local/lib/python3.9/dist-packages (from lazypredict) (1.4.4)
Requirement already satisfied: joblib in /usr/local/lib/python3.9/dist-packages (from lazypredict) (1.1.1)
Requirement already satisfied: lightgbm in /usr/local/lib/python3.9/dist-packages (from lazypredict) (2.2.3)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.9/dist-packages (from lazypredict) (1.2.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.9/dist-packages (from lightgbm->lazypredict) (1.10.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.9/dist-packages (from lightgbm->lazypredict) (1.22.4)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from pandas->lazypredict)
(2022.7.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.9/dist-packages (from pandas-
>lazypredict) (2.8.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from scikit-learn-
>lazypredict) (3.1.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.8.1->pandas-
>lazypredict) (1.15.0)
```

```python
import lazypredict
from lazypredict.Supervised import LazyRegressor
from sklearn.utils import all_estimators
from sklearn.base import RegressorMixin
chosen_regressors = [
    'SVR',
    'BaggingRegressor',
    'RandomForestRegressor',
    'GradientBoostingRegressor',
    'LinearRegression',
    'RidgeCV',
    'LassoCV',
    'KNeighborsRegressor'
]

REGRESSORS = [
    est
    for est in all_estimators()
```

```
    if (issubclass(est[1], RegressorMixin) and (est[0] in chosen_regressors))
]

reg = LazyRegressor(verbose=0, ignore_warnings=False, custom_metric=None,
                    regressors=REGRESSORS)
models, predictions = reg.fit(X_train, X_test, y_train, y_test)
print(models)
```

```
'tuple' object has no attribute '__name__'
Invalid Regressor(s)

100%|██████████| 8/8 [00:03<00:00,  2.23it/s]

                           Adjusted R-Squared  R-Squared       RMSE  \
Model
BaggingRegressor                         0.46       0.46 5373717.07
KNeighborsRegressor                      0.44       0.44 5458492.62
RandomForestRegressor                    0.44       0.44 5474494.94
GradientBoostingRegressor                0.42       0.42 5566706.65
LassoCV                                  0.14       0.14 6781963.84
RidgeCV                                  0.11       0.12 6874607.49
LinearRegression                         0.11       0.11 6879574.63
SVR                                     -0.03      -0.03 7421089.59

                           Time Taken
Model
BaggingRegressor                 0.11
KNeighborsRegressor              0.02
RandomForestRegressor            1.23
GradientBoostingRegressor        0.26
LassoCV                          0.11
RidgeCV                          0.02
LinearRegression                 0.05
SVR                              1.79
```

```
model1 = BaggingRegressor(max_features=1.0,
                          n_estimators=10,
```

```
                          bootstrap=True,
                          random_state=25)
model1.fit(X_train, y_train)

model2 = KNeighborsRegressor(n_neighbors=3)
model2.fit(X_train, y_train)
```

```
▾      KNeighborsRegressor
KNeighborsRegressor(n_neighbors=3)
```

```
y_pred = model1.predict(X_val)
mse = mean_squared_error(y_val, y_pred)
rmse = np.sqrt(mse)

y_pred_2 = model2.predict(X_val)
mse2 = mean_squared_error(y_val, y_pred_2)
rmse2 = np.sqrt(mse2)
```

```
new_data = pd.DataFrame({
    'address/zipcode': ['90024'],
    'bathrooms': [2],
    'bedrooms': [3]
})

price_pred = model1.predict(new_data)
price_pred_2 = model2.predict(new_data)
price_pred, price_pred_2
```

```
(array([2229266.66666667]), array([2389666.66666667]))
```