Objective: Implementing a regression model to analyze Boston housing price with 13 features using Amazon SageMaker.

Approches

1. set up environment(AWS)
2. data exploration
3. data preprocessing
4. data visualization
5. model training using S3 and SageMaker
6. Model evaluation

In [11]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sagemaker
from sagemaker.xgboost.estimator import XGBoost
from sklearn.datasets import fetch_openml
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sagemaker.amazon.linear_learner import LinearLearner
from sklearn.metrics import mean_squared_error, r2_score
import boto3
import os
import logging


# Configure logging
logging.basicConfig(level=logging.INFO)
logging.getLogger('matplotlib').setLevel(logging.ERROR)

logger = logging.getLogger(__name__)

# Set plotting style
sns.set(style="whitegrid")
plt.rcParams['figure.figsize'] = (12, 8)
```

Load the data

In [12]:
```
from sklearn.datasets import fetch_openml

boston = fetch_openml(name="boston", version=1, as_frame=True)
df_X, df_y = boston.data, boston.target
df_boston = pd.concat([df_X, df_y], axis = 1)
df_boston
```

Out[12]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRAT
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	1
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	1
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	1
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	1
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	1
...	...	...	...	...	...	...	...	...	...	...	
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273.0	2
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273.0	2
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273.0	2
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273.0	2
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273.0	2

506 rows × 14 columns

-Variables

There are 14 attributes in each case of the dataset. They are:

CRIM - per capita crime rate by town
ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
INDUS - proportion of non-retail business acres per town.
CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
NOX - nitric oxides concentration (parts per 10 million)
RM - average number of rooms per dwelling
AGE - proportion of owner-occupied units built prior to 1940
DIS - weighted distances to five Boston employment centres
RAD - index of accessibility to radial highways
TAX - full-value property-tax rate per 10,000 dollors
PTRATIO - pupil-teacher ratio by town
B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
LSTAT - % lower status of the population
MEDV - Median value of owner-occupied homes in $1000's

basic exploratory data analysis (EDA)

In [13]:

```
# Display basic information about the dataset
print("Dataset shape:", df_boston.shape)
print("\nFeature descriptions:")
for i, feature in enumerate(boston.feature_names):
    print(f"{feature}: {boston.feature_names[i]}")

print("\nTarget description:")
print(f"MedHouseVal: {boston.target_names[0]}")
```

```
# Display the first few rows of the dataset
print("\nFirst 5 rows of the dataset:")
df_boston.head()
```

```
Dataset shape: (506, 14)

Feature descriptions:
CRIM: CRIM
ZN: ZN
INDUS: INDUS
CHAS: CHAS
NOX: NOX
RM: RM
AGE: AGE
DIS: DIS
RAD: RAD
TAX: TAX
PTRATIO: PTRATIO
B: B
LSTAT: LSTAT

Target description:
MedHouseVal: MEDV

First 5 rows of the dataset:
```

Out[13]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7

In [14]:

```
# Explore the dataset statistics
print("Dataset statistics:")
df_boston.describe()
```

```
Dataset statistics:
```

Out[14]:

	CRIM	ZN	INDUS	NOX	RM	AGE
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.554695	6.284634	68.574901
std	8.601545	23.322453	6.860353	0.115878	0.702617	28.148861
min	0.006320	0.000000	0.460000	0.385000	3.561000	2.900000
25%	0.082045	0.000000	5.190000	0.449000	5.885500	45.025000
50%	0.256510	0.000000	9.690000	0.538000	6.208500	77.500000
75%	3.677083	12.500000	18.100000	0.624000	6.623500	94.075000
max	88.976200	100.000000	27.740000	0.871000	8.780000	100.000000

We can know the standard deviation of ZN, AGE, TAX, and B is higher than other features, so we need further exploration of this data to see whether there're outliers or not. Outliers might influence the result.

In [15]:

```
# Check for missing values
missing_values = df_boston.isnull().sum()
print("Missing values per column:")
print(missing_values)

# If there are no missing values, print a confirmation
if missing_values.sum() == 0:
    print("\nGreat! The dataset has no missing values.")
```

```
Missing values per column:
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64

Great! The dataset has no missing values.
```

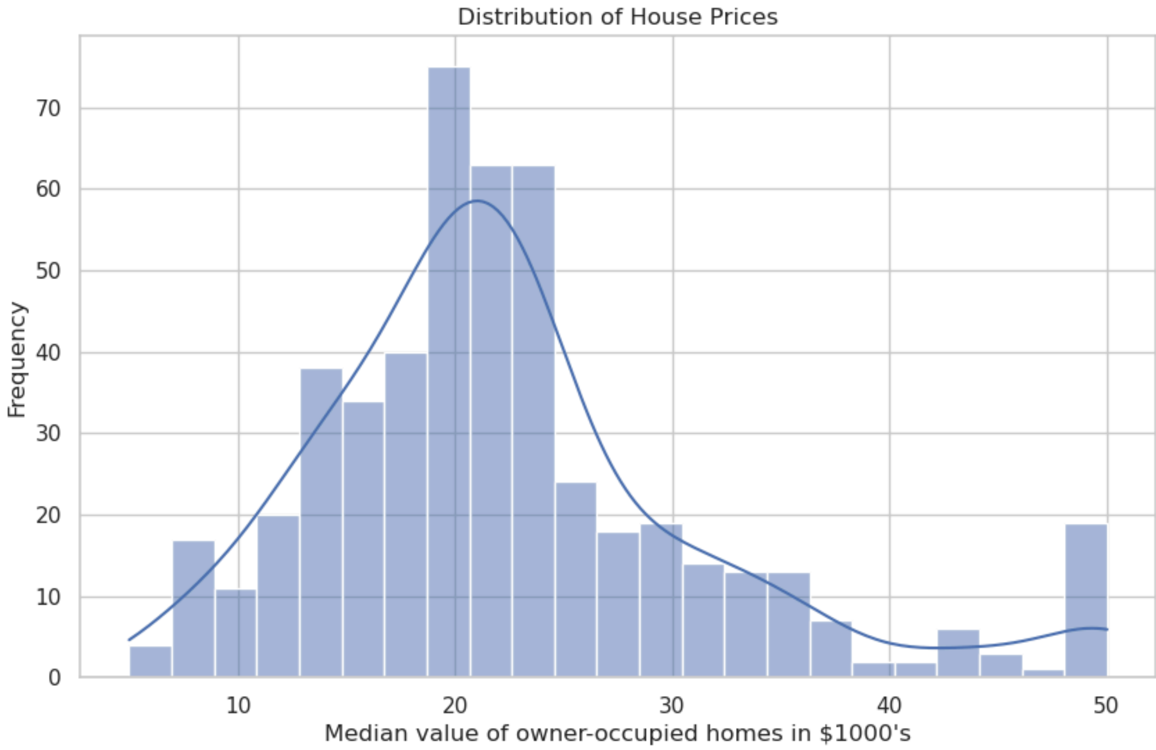
Therefore, we have no need to deal with missing values.

EDA - visuallization

In [16]:

```
# Visualize the distribution of the target variable
plt.figure(figsize=(10, 6))
sns.histplot(df_y, kde=True)
plt.title('Distribution of House Prices')
```

```
plt.xlabel("Median value of owner-occupied homes in $1000's")
plt.ylabel('Frequency')
plt.show()
```

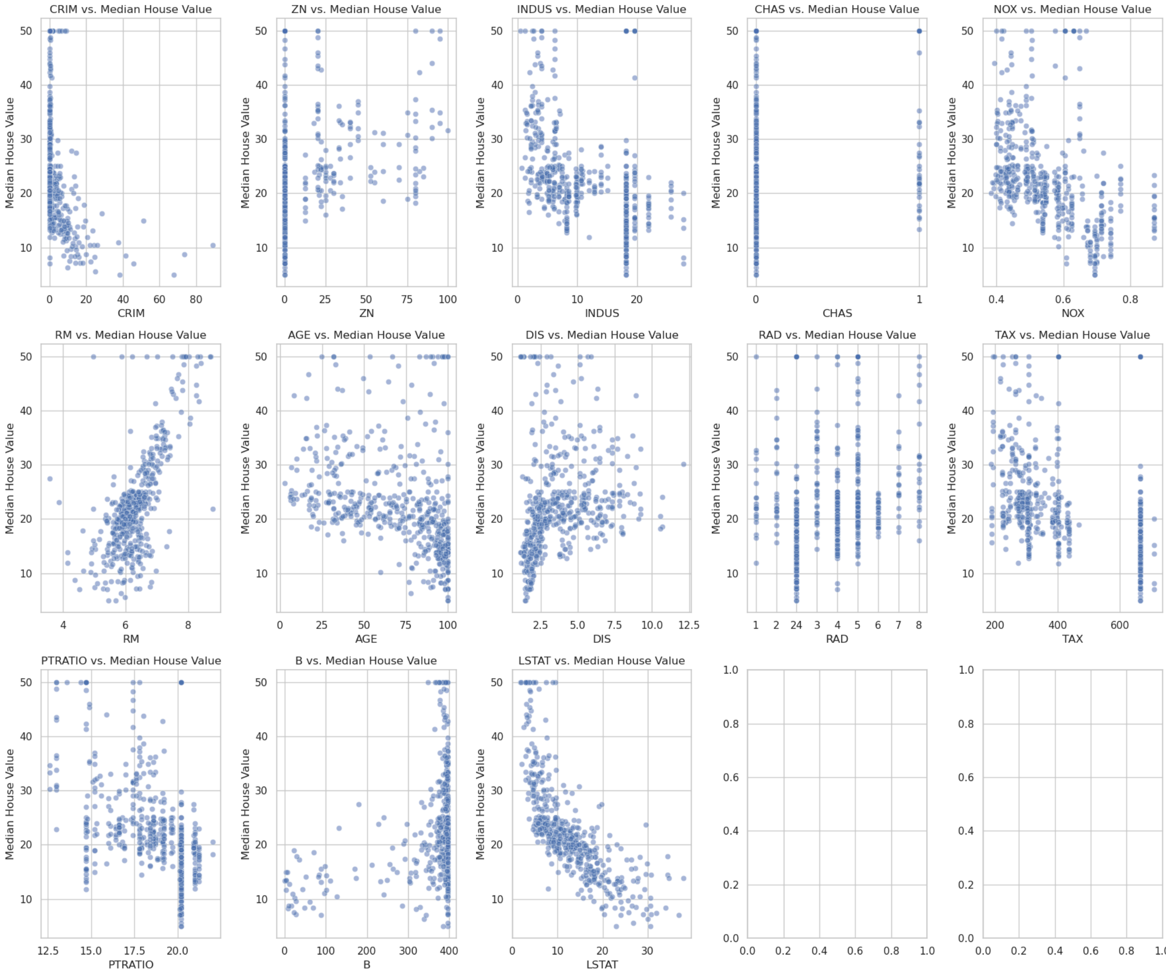


As we can see from aboved picture, the most common median value of owner-occupied homes is 20(in $1000's).

In [17]:
```
# Visualize relationships between key features and the target
fig, axes = plt.subplots(3, 5, figsize=(18, 15))
axes = axes.flatten()

# Plot scatter plots for each feature against the target
for i, feature in enumerate(boston.feature_names):
    if i < len(axes):
        sns.scatterplot(x=df_X[feature], y=df_y, alpha=0.5, ax=axes[i])
        axes[i].set_title(f'{feature} vs. Median House Value')
        axes[i].set_xlabel(feature)
        axes[i].set_ylabel('Median House Value')

plt.tight_layout()
plt.show()
```



Critical Thinking Question: Based on your exploration, which feature do you think might be most strongly related to house prices? Why?

Critical Thinking Answer: RM is likely most correlated with price, because through the aboved picture, I find the picture with RM vs. Median House Value title is high posotive correlated.

In [18]:
```
print("CHAS:", df_X['CHAS'].value_counts())
print("RAD:", df_X['RAD'].value_counts())
```

```
CHAS: 0    471
1     35
Name: CHAS, dtype: int64
RAD: 24    132
5     115
4     110
3      38
6      26
8      24
2      24
1      20
7      17
Name: RAD, dtype: int64
```

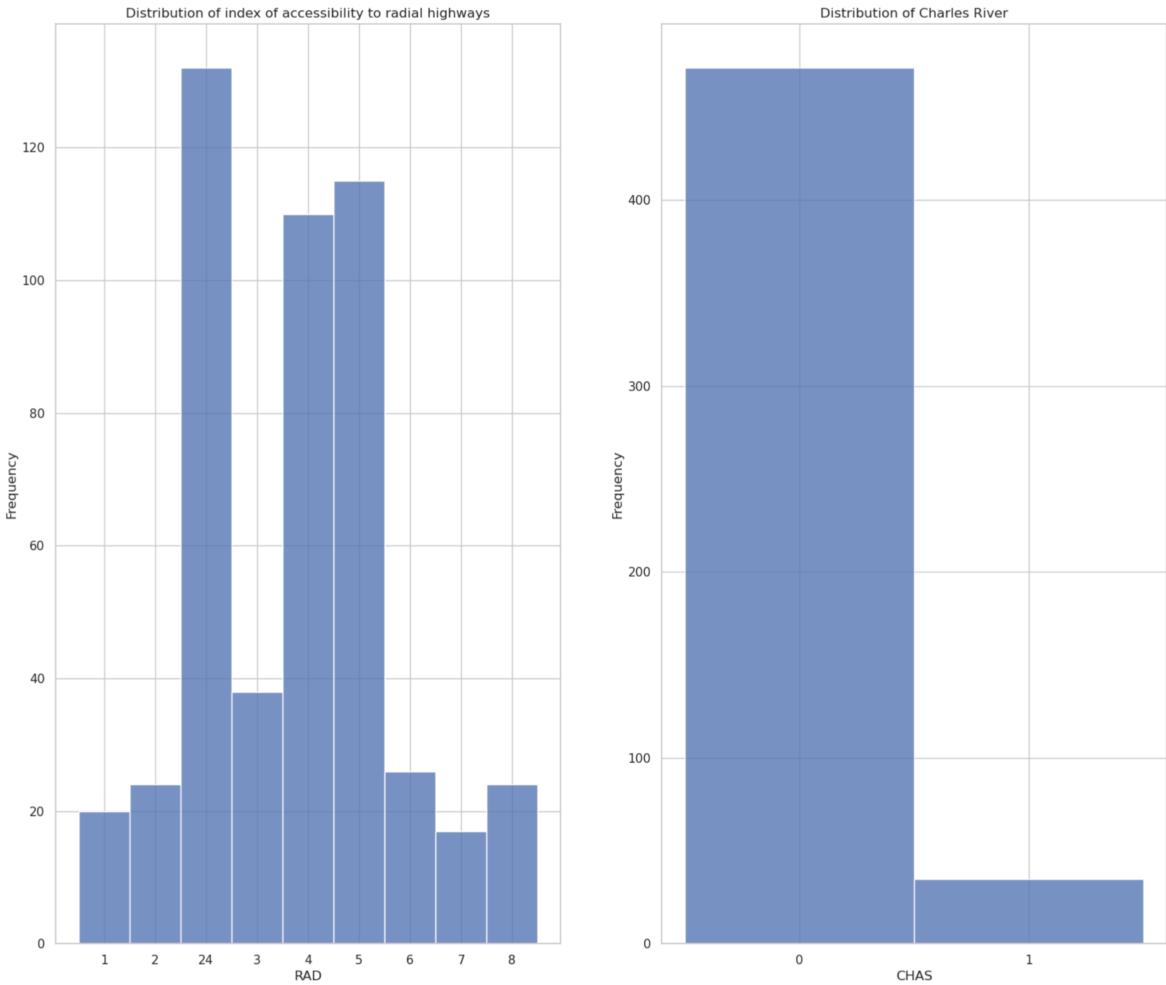
In [19]:
```
# See category distribution
fig, axes = plt.subplots(1, 2, figsize=(18, 15))

sns.histplot(df_X["RAD"], discrete=True, ax=axes[0])
axes[0].set_title("Distribution of index of accessibility to radial highw
```

```
axes[0].set_xlabel("RAD")
axes[0].set_ylabel('Frequency')

axes = axes.flatten()
sns.histplot(df_X["CHAS"], discrete=True, ax=axes[1])
axes[1].set_title("Distribution of Charles River")
axes[1].set_xlabel("CHAS")
axes[1].set_ylabel('Frequency')

plt.show()
```



I find a lot of houses with index of accessibility to radial highways = 24, 4, or 5.

I find houses which tract bounds river is just a few(CHAS=1).

Data Preprocessing

In [20]:
```
X = df_boston.drop('MEDV', axis=1)
y = df_boston['MEDV']
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
y_train = y_train.reset_index(drop=True)
y_test = y_test.reset_index(drop=True)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
X_train_scaled_df = pd.DataFrame(X_train_scaled, columns=X_train.columns)
X_test_scaled_df = pd.DataFrame(X_test_scaled, columns=X_test.columns)
```

```
train_df = X_train_scaled_df.copy()
train_df['MEDV'] = y_train
test_df = X_test_scaled_df.copy()
test_df['MEDV'] = y_test

print(f"Training set shape: {train_df.shape}")
print(f"Testing set shape: {test_df.shape}")

cols = train_df.columns.tolist()
cols.remove('MEDV')
cols = ['MEDV'] + cols

train_df = train_df[cols]
test_df = test_df[cols]
```

```
Training set shape: (404, 14)
Testing set shape: (102, 14)
```

In [12]:
```
# Save the processed data to CSV files
train_df.to_csv('train_xgb.csv', index=False, header=False)
test_df.to_csv('test_xgb.csv', index=False, header=False)
print("Data reformatted for XGBoost and saved to CSV files.")
```

```
Data reformatted for XGBoost and saved to CSV files.
```

In [13]:
```
# Initialize SageMaker session
sagemaker_session = sagemaker.Session()
role = sagemaker.get_execution_role()
bucket = sagemaker_session.default_bucket()
prefix = 'Boston-housing'

# Upload data to S3
train_s3_path = sagemaker_session.upload_data(
    path='train_xgb.csv',
    bucket=bucket,
    key_prefix=f'{prefix}/train'
)

test_s3_path = sagemaker_session.upload_data(
    path='test_xgb.csv',
    bucket=bucket,
    key_prefix=f'{prefix}/test'
)

print(f"Training data uploaded to: {train_s3_path}")
print(f"Testing data uploaded to: {test_s3_path}")
```

```
[03/10/25 07:33:56] INFO     Found credentials from IAM Role:
                             BaseNotebookInstanceEc2InstanceRole
                    INFO     Found credentials from IAM Role:
                             BaseNotebookInstanceEc2InstanceRole
```

```
Training data uploaded to: s3://sagemaker-us-east-1-181786711311/Boston-ho
using/train/train_xgb.csv
Testing data uploaded to: s3://sagemaker-us-east-1-181786711311/Boston-hou
sing/test/test_xgb.csv
```

Critical Thinking Question: Why is scaling features important for many machine learning algorithms? How might unscaled features affect model performance?

Critical Thinking Answer: Scaling is important to ensure numerical stability and faster convergence.

Model Training with SageMaker

In [14]:
```
# Configure the XGBoost estimator
xgb = XGBoost(
    entry_point='script.py',
    framework_version='1.5-1',
    hyperparameters={
        'objective': 'reg:squarederror',
        'max_depth': 6,
        'eta': 0.1,
        'gamma': 5,
        'min_child_weight': 10,
        'subsample': 0.8,
        'colsample_bytree': 0.8,
        'verbosity': 1,
        'num_round': 100
    },
    role=role,
    instance_count=1,
    instance_type='ml.m5.xlarge',
    output_path=f's3://{bucket}/{prefix}/output'
)

# Create a training script
with open('script.py', 'w') as f:
    f.write("""
import argparse
import os
import json
import pandas as pd
import numpy as np
import xgboost as xgb
from sklearn.metrics import mean_squared_error

def model_fn(model_dir):
    model = xgb.Booster()
    model.load_model(os.path.join(model_dir, 'xgboost-model'))
    return model

if __name__ == '__main__':
    parser = argparse.ArgumentParser()

    # Hyperparameters
    parser.add_argument('--objective', type=str, default='reg:squarederro
    parser.add_argument('--max_depth', type=int, default=5)
    parser.add_argument('--eta', type=float, default=0.2)
    parser.add_argument('--gamma', type=float, default=4)
    parser.add_argument('--min_child_weight', type=int, default=6)
    parser.add_argument('--subsample', type=float, default=0.8)
    parser.add_argument('--verbosity', type=int, default=1)
    parser.add_argument('--num_round', type=int, default=50)

    # SageMaker parameters
    parser.add_argument('--model_dir', type=str, default=os.environ.get('
```

```
    parser.add_argument('--train', type=str, default=os.environ.get('SM_C
    parser.add_argument('--validation', type=str, default=os.environ.get(

    args, _ = parser.parse_known_args()

    # Load data
    train_data = pd.read_csv(os.path.join(args.train, 'train_xgb.csv'), h
    validation_data = pd.read_csv(os.path.join(args.validation, 'test_xgb

    # Split into features and target
    X_train = train_data.iloc[:, 1:].values
    y_train = train_data.iloc[:, 0].values
    X_validation = validation_data.iloc[:, 1:].values
    y_validation = validation_data.iloc[:, 0].values

    # Create DMatrix
    dtrain = xgb.DMatrix(X_train, label=y_train)
    dvalidation = xgb.DMatrix(X_validation, label=y_validation)

    # Train model
    params = {
        'objective': args.objective,
        'max_depth': args.max_depth,
        'eta': args.eta,
        'gamma': args.gamma,
        'min_child_weight': args.min_child_weight,
        'subsample': args.subsample,
        'verbosity': args.verbosity
    }

    watchlist = [(dtrain, 'train'), (dvalidation, 'validation')]

    model = xgb.train(
        params=params,
        dtrain=dtrain,
        num_boost_round=args.num_round,
        evals=watchlist
    )

    # Evaluate model
    predictions = model.predict(dvalidation)
    rmse = np.sqrt(mean_squared_error(y_validation, predictions))
    print(f'Validation RMSE: {rmse:.4f}')

    # Save model
    model.save_model(os.path.join(args.model_dir, 'xgboost-model'))

    # Save feature importance
    feature_importance = model.get_score(importance_type='weight')
    with open(os.path.join(args.model_dir, 'feature_importance.json'), 'w
        json.dump(feature_importance, f)
""")

print("Training script created.")
```

```
[03/10/25 07:33:58] INFO     Found credentials from IAM Role:
                             BaseNotebookInstanceEc2InstanceRole
```

```
                 INFO     Found credentials from IAM Role:
                          BaseNotebookInstanceEc2InstanceRole
```

```
                 INFO     Ignoring unnecessary Python version: py
```

```
[03/10/25 07:33:59] INFO     Ignoring unnecessary instance type: ml.
```

```
Training script created.
```

In [15]:

```
xgb.fit({
    'train': train_s3_path,
    'validation': test_s3_path
})

print("Model training completed!")
```

```
                 INFO     SageMaker Python SDK will collect teler
                          understand our user's needs, diagnose i
                          additional features.
                          To opt out of telemetry, please disable
                          parameter in SDK defaults config. For r
                          to
                          https://sagemaker.readthedocs.io/en/sta
                          guring-and-using-defaults-with-the-sage
```

```
                 INFO     Creating training-job with name:
                          sagemaker-xgboost-2025-03-10-07-33-59-(
```

```
2025-03-10 07:34:02 Starting - Starting the training job...
..25-03-10 07:34:15 Starting - Preparing the instances for training.
.....03-10 07:35:02 Downloading - Downloading the training image.
2025-03-10 07:36:04 Training - Training image download completed. Training
in progress.
2025-03-10 07:36:04 Uploading - Uploading generated training model/minicon
da3/lib/python3.8/site-packages/xgboost/compat.py:36: FutureWarning: panda
s.Int64Index is deprecated and will be removed from pandas in a future ver
sion. Use pandas.Index with the appropriate dtype instead.
  from pandas import MultiIndex, Int64Index
[2025-03-10 07:35:55.185 ip-10-0-174-208.ec2.internal:7 INFO utils.py:28]
RULE_JOB_STOP_SIGNAL_FILENAME: None
[2025-03-10 07:35:55.207 ip-10-0-174-208.ec2.internal:7 INFO profiler_conf
ig_parser.py:111] User has disabled profiler.
[2025-03-10:07:35:55:INFO] Imported framework sagemaker_xgboost_container.
training
[2025-03-10:07:35:55:INFO] No GPUs detected (normal if no gpus installed)
[2025-03-10:07:35:55:INFO] Invoking user training script.
[2025-03-10:07:35:55:INFO] Module script does not provide a setup.py.
Generating setup.py
[2025-03-10:07:35:55:INFO] Generating setup.cfg
[2025-03-10:07:35:55:INFO] Generating MANIFEST.in
[2025-03-10:07:35:55:INFO] Installing module with the following command:
/miniconda3/bin/python3 -m pip install .
Processing /opt/ml/code
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Building wheels for collected packages: script
  Building wheel for script (setup.py): started
  Building wheel for script (setup.py): finished with status 'done'
  Created wheel for script: filename=script-1.0.0-py2.py3-none-any.whl siz
e=4508 sha256=c6083fcaa253f0e11f4c6b94c03512902b53f94e78dec04d2ee8b4562c94
2785
  Stored in directory: /home/model-server/tmp/pip-ephem-wheel-cache-elde22
xw/wheels/f3/75/57/158162e9eab7af12b5c338c279b3a81f103b89d74eeb911c00
Successfully built script
Installing collected packages: script
Successfully installed script-1.0.0
WARNING: Running pip as the 'root' user can result in broken permissions a
nd conflicting behaviour with the system package manager. It is recommende
d to use a virtual environment instead: https://pip.pypa.io/warnings/venv
[2025-03-10:07:35:57:INFO] No GPUs detected (normal if no gpus installed)
[2025-03-10:07:35:57:INFO] Invoking user script
Training Env:
{
    "additional_framework_parameters": {},
    "channel_input_dirs": {
        "train": "/opt/ml/input/data/train",
        "validation": "/opt/ml/input/data/validation"
    },
    "current_host": "algo-1",
    "framework_module": "sagemaker_xgboost_container.training:main",
    "hosts": [
        "algo-1"
    ],
    "hyperparameters": {
        "colsample_bytree": 0.8,
        "eta": 0.1,
        "gamma": 5,
        "max_depth": 6,
```

```
        "min_child_weight": 10,
        "num_round": 100,
        "objective": "reg:squarederror",
        "subsample": 0.8,
        "verbosity": 1
    },
    "input_config_dir": "/opt/ml/input/config",
    "input_data_config": {
        "train": {
            "TrainingInputMode": "File",
            "S3DistributionType": "FullyReplicated",
            "RecordWrapperType": "None"
        },
        "validation": {
            "TrainingInputMode": "File",
            "S3DistributionType": "FullyReplicated",
            "RecordWrapperType": "None"
        }
    },
    "input_dir": "/opt/ml/input",
    "is_master": true,
    "job_name": "sagemaker-xgboost-2025-03-10-07-33-59-093",
    "log_level": 20,
    "master_hostname": "algo-1",
    "model_dir": "/opt/ml/model",
    "module_dir": "s3://sagemaker-us-east-1-181786711311/sagemaker-xgboost
-2025-03-10-07-33-59-093/source/sourcedir.tar.gz",
    "module_name": "script",
    "network_interface_name": "eth0",
    "num_cpus": 4,
    "num_gpus": 0,
    "output_data_dir": "/opt/ml/output/data",
    "output_dir": "/opt/ml/output",
    "output_intermediate_dir": "/opt/ml/output/intermediate",
    "resource_config": {
        "current_host": "algo-1",
        "current_instance_type": "ml.m5.xlarge",
        "current_group_name": "homogeneousCluster",
        "hosts": [
            "algo-1"
        ],
        "instance_groups": [
            {
                "instance_group_name": "homogeneousCluster",
                "instance_type": "ml.m5.xlarge",
                "hosts": [
                    "algo-1"
                ]
            }
        ],
        "network_interface_name": "eth0"
    },
    "user_entry_point": "script.py"
}
Environment variables:
SM_HOSTS=["algo-1"]
SM_NETWORK_INTERFACE_NAME=eth0
SM_HPS={"colsample_bytree":0.8,"eta":0.1,"gamma":5,"max_depth":6,"min_chil
d_weight":10,"num_round":100,"objective":"reg:squarederror","subsample":0.
8,"verbosity":1}
```

```
SM_USER_ENTRY_POINT=script.py
SM_FRAMEWORK_PARAMS={}
SM_RESOURCE_CONFIG={"current_group_name":"homogeneousCluster","current_hos
t":"algo-1","current_instance_type":"ml.m5.xlarge","hosts":["algo-1"],"ins
tance_groups":[{"hosts":["algo-1"],"instance_group_name":"homogeneousClust
er","instance_type":"ml.m5.xlarge"}],"network_interface_name":"eth0"}
SM_INPUT_DATA_CONFIG={"train":{"RecordWrapperType":"None","S3DistributionT
ype":"FullyReplicated","TrainingInputMode":"File"},"validation":{"RecordWr
apperType":"None","S3DistributionType":"FullyReplicated","TrainingInputMod
e":"File"}}
SM_OUTPUT_DATA_DIR=/opt/ml/output/data
SM_CHANNELS=["train","validation"]
SM_CURRENT_HOST=algo-1
SM_MODULE_NAME=script
SM_LOG_LEVEL=20
SM_FRAMEWORK_MODULE=sagemaker_xgboost_container.training:main
SM_INPUT_DIR=/opt/ml/input
SM_INPUT_CONFIG_DIR=/opt/ml/input/config
SM_OUTPUT_DIR=/opt/ml/output
SM_NUM_CPUS=4
SM_NUM_GPUS=0
SM_MODEL_DIR=/opt/ml/model
SM_MODULE_DIR=s3://sagemaker-us-east-1-181786711311/sagemaker-xgboost-2025
-03-10-07-33-59-093/source/sourcedir.tar.gz
SM_TRAINING_ENV={"additional_framework_parameters":{},"channel_input_dir
s":{"train":"/opt/ml/input/data/train","validation":"/opt/ml/input/data/va
lidation"},"current_host":"algo-1","framework_module":"sagemaker_xgboost_c
ontainer.training:main","hosts":["algo-1"],"hyperparameters":{"colsample_b
ytree":0.8,"eta":0.1,"gamma":5,"max_depth":6,"min_child_weight":10,"num_ro
und":100,"objective":"reg:squarederror","subsample":0.8,"verbosity":1},"in
put_config_dir":"/opt/ml/input/config","input_data_config":{"train":{"Reco
rdWrapperType":"None","S3DistributionType":"FullyReplicated","TrainingInpu
tMode":"File"},"validation":{"RecordWrapperType":"None","S3DistributionTyp
e":"FullyReplicated","TrainingInputMode":"File"}},"input_dir":"/opt/ml/inp
ut","is_master":true,"job_name":"sagemaker-xgboost-2025-03-10-07-33-59-09
3","log_level":20,"master_hostname":"algo-1","model_dir":"/opt/ml/mode
l","module_dir":"s3://sagemaker-us-east-1-181786711311/sagemaker-xgboost-2
025-03-10-07-33-59-093/source/sourcedir.tar.gz","module_name":"script","ne
twork_interface_name":"eth0","num_cpus":4,"num_gpus":0,"output_data_di
r":"/opt/ml/output/data","output_dir":"/opt/ml/output","output_intermediat
e_dir":"/opt/ml/output/intermediate","resource_config":{"current_group_nam
e":"homogeneousCluster","current_host":"algo-1","current_instance_type":"m
l.m5.xlarge","hosts":["algo-1"],"instance_groups":[{"hosts":["algo-1"],"in
stance_group_name":"homogeneousCluster","instance_type":"ml.m5.xlarg
e"}],"network_interface_name":"eth0"},"user_entry_point":"script.py"}
SM_USER_ARGS=["--colsample_bytree","0.8","--eta","0.1","--gamma","5","--ma
x_depth","6","--min_child_weight","10","--num_round","100","--objectiv
e","reg:squarederror","--subsample","0.8","--verbosity","1"]
SM_OUTPUT_INTERMEDIATE_DIR=/opt/ml/output/intermediate
SM_CHANNEL_TRAIN=/opt/ml/input/data/train
SM_CHANNEL_VALIDATION=/opt/ml/input/data/validation
SM_HP_COLSAMPLE_BYTREE=0.8
SM_HP_ETA=0.1
SM_HP_GAMMA=5
SM_HP_MAX_DEPTH=6
SM_HP_MIN_CHILD_WEIGHT=10
SM_HP_NUM_ROUND=100
SM_HP_OBJECTIVE=reg:squarederror
SM_HP_SUBSAMPLE=0.8
SM_HP_VERBOSITY=1
```

```
PYTHONPATH=/miniconda3/bin:/:/miniconda3/lib/python/site-packages/xgboost/
dmlc-core/tracker:/miniconda3/lib/python38.zip:/miniconda3/lib/python3.8:/
miniconda3/lib/python3.8/lib-dynload:/miniconda3/lib/python3.8/site-packag
es
Invoking script with the following command:
/miniconda3/bin/python3 -m script --colsample_bytree 0.8 --eta 0.1 --gamma
5 --max_depth 6 --min_child_weight 10 --num_round 100 --objective reg:squa
rederror --subsample 0.8 --verbosity 1
/miniconda3/lib/python3.8/site-packages/xgboost/compat.py:36: FutureWarnin
g: pandas.Int64Index is deprecated and will be removed from pandas in a fu
ture version. Use pandas.Index with the appropriate dtype instead.
  from pandas import MultiIndex, Int64Index
[0]#011train-rmse:21.89355#011validation-rmse:20.57107
[1]#011train-rmse:19.84149#011validation-rmse:18.66716
[2]#011train-rmse:17.99840#011validation-rmse:16.96808
[3]#011train-rmse:16.33244#011validation-rmse:15.44099
[4]#011train-rmse:14.86379#011validation-rmse:14.08338
[5]#011train-rmse:13.53011#011validation-rmse:12.83068
[6]#011train-rmse:12.34113#011validation-rmse:11.74271
[7]#011train-rmse:11.28243#011validation-rmse:10.78188
[8]#011train-rmse:10.30809#011validation-rmse:9.92938
[9]#011train-rmse:9.41050#011validation-rmse:9.15246
[10]#011train-rmse:8.62197#011validation-rmse:8.41572
[11]#011train-rmse:7.90970#011validation-rmse:7.80373
[12]#011train-rmse:7.26885#011validation-rmse:7.29514
[13]#011train-rmse:6.69991#011validation-rmse:6.79954
[14]#011train-rmse:6.19835#011validation-rmse:6.36149
[15]#011train-rmse:5.73026#011validation-rmse:6.00833
[16]#011train-rmse:5.31114#011validation-rmse:5.64493
[17]#011train-rmse:4.94000#011validation-rmse:5.35240
[18]#011train-rmse:4.59113#011validation-rmse:5.10095
[19]#011train-rmse:4.28953#011validation-rmse:4.90531
[20]#011train-rmse:4.02848#011validation-rmse:4.72927
[21]#011train-rmse:3.78928#011validation-rmse:4.54990
[22]#011train-rmse:3.59288#011validation-rmse:4.42519
[23]#011train-rmse:3.42075#011validation-rmse:4.29206
[24]#011train-rmse:3.26220#011validation-rmse:4.18140
[25]#011train-rmse:3.12173#011validation-rmse:4.06366
[26]#011train-rmse:2.99068#011validation-rmse:3.97541
[27]#011train-rmse:2.87604#011validation-rmse:3.91881
[28]#011train-rmse:2.78153#011validation-rmse:3.84900
[29]#011train-rmse:2.67665#011validation-rmse:3.79770
[30]#011train-rmse:2.60499#011validation-rmse:3.76643
[31]#011train-rmse:2.54385#011validation-rmse:3.72113
[32]#011train-rmse:2.47907#011validation-rmse:3.68114
[33]#011train-rmse:2.42033#011validation-rmse:3.64939
[34]#011train-rmse:2.36937#011validation-rmse:3.61988
[35]#011train-rmse:2.31643#011validation-rmse:3.55517
[36]#011train-rmse:2.28116#011validation-rmse:3.53870
[37]#011train-rmse:2.24847#011validation-rmse:3.51215
[38]#011train-rmse:2.19830#011validation-rmse:3.50122
[39]#011train-rmse:2.17668#011validation-rmse:3.48819
[40]#011train-rmse:2.15166#011validation-rmse:3.47572
[41]#011train-rmse:2.12505#011validation-rmse:3.47369
[42]#011train-rmse:2.09825#011validation-rmse:3.44745
[43]#011train-rmse:2.06700#011validation-rmse:3.43257
[44]#011train-rmse:2.03255#011validation-rmse:3.43077
[45]#011train-rmse:2.01309#011validation-rmse:3.42781
[46]#011train-rmse:1.99673#011validation-rmse:3.41285
[47]#011train-rmse:1.97659#011validation-rmse:3.39735
```

```
[48]#011train-rmse:1.95147#011validation-rmse:3.36520
[49]#011train-rmse:1.92757#011validation-rmse:3.35217
[50]#011train-rmse:1.90434#011validation-rmse:3.32232
[51]#011train-rmse:1.89392#011validation-rmse:3.31373
[52]#011train-rmse:1.87453#011validation-rmse:3.30754
[53]#011train-rmse:1.85286#011validation-rmse:3.28055
[54]#011train-rmse:1.83970#011validation-rmse:3.27139
[55]#011train-rmse:1.81299#011validation-rmse:3.25046
[56]#011train-rmse:1.80315#011validation-rmse:3.24513
[57]#011train-rmse:1.78525#011validation-rmse:3.22134
[58]#011train-rmse:1.77211#011validation-rmse:3.21776
[59]#011train-rmse:1.75886#011validation-rmse:3.21558
[60]#011train-rmse:1.74964#011validation-rmse:3.21862
[61]#011train-rmse:1.73361#011validation-rmse:3.21483
[62]#011train-rmse:1.71863#011validation-rmse:3.21481
[63]#011train-rmse:1.69984#011validation-rmse:3.19447
[64]#011train-rmse:1.68727#011validation-rmse:3.19426
[65]#011train-rmse:1.67668#011validation-rmse:3.19031
[66]#011train-rmse:1.66630#011validation-rmse:3.20017
[67]#011train-rmse:1.66165#011validation-rmse:3.20100
[68]#011train-rmse:1.64851#011validation-rmse:3.18940
[69]#011train-rmse:1.63026#011validation-rmse:3.18299
[70]#011train-rmse:1.62351#011validation-rmse:3.18442
[71]#011train-rmse:1.60926#011validation-rmse:3.18527
[72]#011train-rmse:1.59776#011validation-rmse:3.18675
[73]#011train-rmse:1.58781#011validation-rmse:3.18249
[74]#011train-rmse:1.57766#011validation-rmse:3.18002
[75]#011train-rmse:1.56672#011validation-rmse:3.17107
[76]#011train-rmse:1.55880#011validation-rmse:3.15989
[77]#011train-rmse:1.54599#011validation-rmse:3.15761
[78]#011train-rmse:1.53484#011validation-rmse:3.14435
[79]#011train-rmse:1.52909#011validation-rmse:3.14123
[80]#011train-rmse:1.51628#011validation-rmse:3.13793
[81]#011train-rmse:1.50723#011validation-rmse:3.12476
[82]#011train-rmse:1.48989#011validation-rmse:3.13047
[83]#011train-rmse:1.48298#011validation-rmse:3.13093
[84]#011train-rmse:1.47709#011validation-rmse:3.13526
[85]#011train-rmse:1.47271#011validation-rmse:3.13606
[86]#011train-rmse:1.46343#011validation-rmse:3.12836
[87]#011train-rmse:1.45229#011validation-rmse:3.12822
[88]#011train-rmse:1.44281#011validation-rmse:3.11873
[89]#011train-rmse:1.42609#011validation-rmse:3.10369
[90]#011train-rmse:1.42171#011validation-rmse:3.10451
[91]#011train-rmse:1.41283#011validation-rmse:3.10843
[92]#011train-rmse:1.39466#011validation-rmse:3.11284
[93]#011train-rmse:1.38788#011validation-rmse:3.10957
[94]#011train-rmse:1.38112#011validation-rmse:3.10785
[95]#011train-rmse:1.37228#011validation-rmse:3.10469
[96]#011train-rmse:1.36411#011validation-rmse:3.10454
[97]#011train-rmse:1.35753#011validation-rmse:3.10023
[98]#011train-rmse:1.35260#011validation-rmse:3.10010
[99]#011train-rmse:1.34560#011validation-rmse:3.10257
Validation RMSE: 3.1026

2025-03-10 07:36:17 Completed - Training job completed
Training seconds: 99
Billable seconds: 99
Model training completed!
```

Critical Thinking Question: What are the advantages of training a model in the cloud (like SageMaker) compared to training locally on your computer?

Critical Thinking Answer: Cloud training allows scalability, avoids local hardware limitations, and simplifies deployment. In fact, you can also link to S3 to ensure data reliability.

In [22]:
```
model_data = xgb.model_data  # getting SageMaker XGBoost Estimator from S

model_path = model_data.replace('s3://', '')
bucket_name = model_path.split('/')[0]
key = '/'.join(model_path.split('/')[1:])

s3 = boto3.client('s3')
s3.download_file(bucket_name, key, 'model.tar.gz')

# Extract the model artifacts
import tarfile
with tarfile.open('model.tar.gz') as tar:
    tar.extractall(path='model')

# Load the model locally
!pip install xgboost
import xgboost as xgb
model = xgb.Booster()
model.load_model('/home/ec2-user/SageMaker/model/xgboost-model')

# Make predictions locally
dtest = xgb.DMatrix(X_test.values)
predictions = model.predict(dtest)

# Evaluate the model
mse = mean_squared_error(y_test, predictions)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, predictions)

print(f"Mean Squared Error: {mse:.4f}")
print(f"Root Mean Squared Error: {rmse:.4f}")
print(f"R² Score: {r2:.4f}")
```

```
Requirement already satisfied: xgboost in /home/ec2-user/anaconda3/envs/py
thon3/lib/python3.10/site-packages (2.1.4)
Requirement already satisfied: numpy in /home/ec2-user/anaconda3/envs/pyth
on3/lib/python3.10/site-packages (from xgboost) (1.26.4)
Requirement already satisfied: scipy in /home/ec2-user/anaconda3/envs/pyth
on3/lib/python3.10/site-packages (from xgboost) (1.15.1)
Mean Squared Error: 97.9597
Root Mean Squared Error: 9.8975
R² Score: -0.3358
```

Critical Thinking Question: Based on your evaluation metrics, how well did your model perform? What might explain any discrepancies between predicted and actual values?
The following are both Thoughtful analysis of model performance and response to critical thinking question

The result of this model is terrible.

1. Mean Squared Error (MSE): 97.96. and Root Mean Squared Error (RMSE): 9.90. High MSE and RMSE suggests large errors in predictions.
2. R^2 Score: -0.3358. R^2 is negative, indicating that the model performs worse than a simple mean predictor

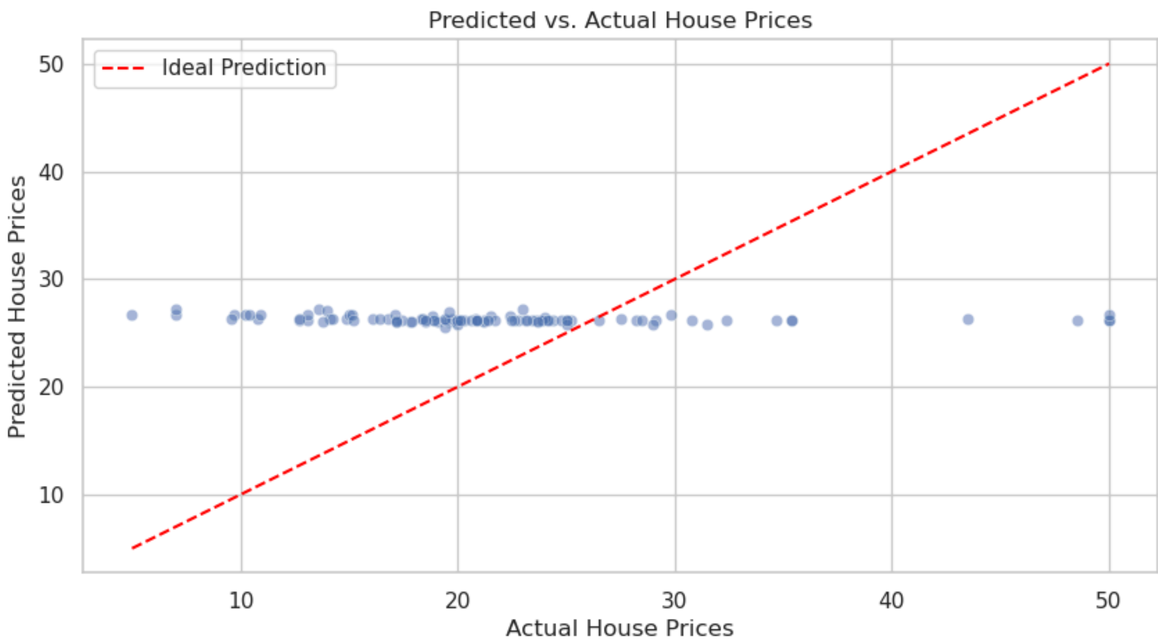
Discrepancies between predicted and actual values may because:

1. The dataset may contain outliers or an uneven distribution of values, which could distort predictions.
2. XGBoost 'reg:squarederror' works well with linear data, but if the relationship between features and house prices is highly non-linear, additional feature engineering or preprocessing might be needed.
3. I did a bad job on tuning the model.

In [24]:
```
# Scatter plot: Predicted vs. Actual House Prices
plt.figure(figsize=(10, 5))
sns.scatterplot(x=y_test, y=predictions, alpha=0.5)

# Plot a reference line (ideal prediction)
min_val = min(y_test.min(), predictions.min())
max_val = max(y_test.max(), predictions.max())
plt.plot([min_val, max_val], [min_val, max_val], color='red', linestyle='

# Labels and title
plt.xlabel("Actual House Prices")
plt.ylabel("Predicted House Prices")
plt.title("Predicted vs. Actual House Prices")
plt.legend()
plt.show()
```



Through the scatter plot, we can also know this model prediction perform badly, because while Actual house prices are various, predicted house price is always around 28.