In [1]:
```
pip install xgboost numpy pandas scikit-learn matplotlib seaborn
```

```
Note: you may need to restart the kernel to use updated packages.Collecting xg
boost
  Obtaining dependency information for xgboost from https://files.pythonhoste
d.org/packages/24/ec/ad387100fa3cc2b9b81af0829b5ecfe75ec5bb19dd7c19d4fea06fb81
802/xgboost-2.0.3-py3-none-win_amd64.whl.metadata (https://files.pythonhosted.
org/packages/24/ec/ad387100fa3cc2b9b81af0829b5ecfe75ec5bb19dd7c19d4fea06fb8180
2/xgboost-2.0.3-py3-none-win_amd64.whl.metadata)
  Downloading xgboost-2.0.3-py3-none-win_amd64.whl.metadata (2.0 kB)
Requirement already satisfied: numpy in c:\users\aditya kudva\anaconda3\lib\si
te-packages (1.24.3)
Requirement already satisfied: pandas in c:\users\aditya kudva\anaconda3\lib\s
ite-packages (1.5.3)
Requirement already satisfied: scikit-learn in c:\users\aditya kudva\anaconda3
\lib\site-packages (1.3.0)
Requirement already satisfied: matplotlib in c:\users\aditya kudva\anaconda3\l
ib\site-packages (3.7.1)
Requirement already satisfied: seaborn in c:\users\aditya kudva\anaconda3\lib
\site-packages (0.12.2)
Requirement already satisfied: scipy in c:\users\aditya kudva\anaconda3\lib\si
```

In [3]:
```
import xgboost as xgb
import pandas as pd
import numpy as np
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```python
data = fetch_california_housing()
data
```

Out[4]:
```
{'data': array([[   8.3252    ,   41.        ,    6.98412698, ...,    2.5555555
6,
          37.88      , -122.23      ],
        [   8.3014    ,   21.        ,    6.23813708, ...,    2.10984183,
          37.86      , -122.22      ],
        [   7.2574    ,   52.        ,    8.28813559, ...,    2.80225989,
          37.85      , -122.24      ],
        ...,
        [   1.7       ,   17.        ,    5.20554273, ...,    2.3256351 ,
          39.43      , -121.22      ],
        [   1.8672    ,   18.        ,    5.32951289, ...,    2.12320917,
          39.43      , -121.32      ],
        [   2.3886    ,   16.        ,    5.25471698, ...,    2.61698113,
          39.37      , -121.24      ]]),
 'target': array([4.526, 3.585, 3.521, ..., 0.923, 0.847, 0.894]),
 'frame': None,
 'target_names': ['MedHouseVal'],
 'feature_names': ['MedInc',
  'HouseAge',
  'AveRooms',
  'AveBedrms',
  'Population',
  'AveOccup',
  'Latitude',
  'Longitude'],
 'DESCR': '.. _california_housing_dataset:\n\nCalifornia Housing dataset\n------
--------------------\n\n**Data Set Characteristics:**\n\n    :Number of Instance
s: 20640\n\n    :Number of Attributes: 8 numeric, predictive attributes and the
target\n\n    :Attribute Information:\n        - MedInc        median income in
block group\n        - HouseAge      median house age in block group\n        -
AveRooms      average number of rooms per household\n        - AveBedrms     ave
rage number of bedrooms per household\n        - Population    block group popul
ation\n        - AveOccup      average number of household members\n        - La
titude      block group latitude\n        - Longitude     block group longitude
\n\n    :Missing Attribute Values: None\n\nThis dataset was obtained from the St
atLib repository.\nhttps://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html
\n\nThe target variable is the median house value for California districts,\nexp
ressed in hundreds of thousands of dollars ($100,000).\n\nThis dataset was deriv
ed from the 1990 U.S. census, using one row per census\nblock group. A block gro
up is the smallest geographical unit for which the U.S.\nCensus Bureau publishes
sample data (a block group typically has a population\nof 600 to 3,000 peopl
e).\n\nA household is a group of people residing within a home. Since the averag
e\nnumber of rooms and bedrooms in this dataset are provided per household, thes
e\ncolumns may take surprisingly large values for block groups with few househol
ds\nand many empty houses, such as vacation resorts.\n\nIt can be downloaded/loa
ded using the\n:func:`sklearn.datasets.fetch_california_housing` function.\n\n..
topic:: References\n\n    - Pace, R. Kelley and Ronald Barry, Sparse Spatial Aut
oregressions,\n      Statistics and Probability Letters, 33 (1997) 291-297\n'}
```

In [5]:
```
X = pd.DataFrame(data.data, columns=data.feature_names)
X
```

Out[5]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25
...	...	...	...	...	...	...	...	...
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	-121.09
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	-121.21
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	-121.22
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	-121.32
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	-121.24

20640 rows × 8 columns

In [6]:
```
y = pd.Series(data.target)
y
```

Out[6]:
```
0        4.526
1        3.585
2        3.521
3        3.413
4        3.422
         ...
20635    0.781
20636    0.771
20637    0.923
20638    0.847
20639    0.894
Length: 20640, dtype: float64
```

In [7]:
```
X['Income^2'] = X['MedInc'] ** 2
X['Age*Rooms'] = X['HouseAge'] * X['AveRooms']
```

In [8]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X = pd.DataFrame(X_scaled, columns=X.columns)
```

In [9]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_s
X_train, X_test, y_train, y_test
```

```
Out[9]: (          MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  \
 14196 -0.321654  0.346478 -0.166259  -0.190451    0.772251  0.059808
 8267  -0.030620  1.617807 -0.386181  -0.117472   -0.098440 -0.128306
 17445  0.150349 -1.957806  0.087641  -0.235400   -0.450778 -0.033453
 14265 -1.014947  0.584852 -0.576442  -0.132670   -0.006602  0.088940
 2271  -0.166583  1.141059  0.339282   0.079205   -0.486983 -0.074203
 ...         ...       ...       ...        ...         ...       ...
 11284  1.315592  0.505394  0.282943  -0.359587   -0.677723 -0.003697
 11964 -0.431983  0.346478  0.581864   0.364661    0.289220  0.080261
 5390  -0.492832  0.584852 -0.582949  -0.035828    0.291870  0.025170
 860    0.973025 -1.083767  0.390584  -0.060554    0.310414  0.010422
 15795 -0.681749  1.856182 -0.819050  -0.079973    1.053944 -0.092623

        Latitude  Longitude  Income^2  Age*Rooms
 14196 -1.367976   1.267645 -0.362993   0.169001
 8267  -0.871699   0.703627 -0.184821   0.778354
 17445 -0.455012  -0.454356 -0.059984  -1.456053
 14265 -1.377340   1.227714 -0.675199  -0.075113
 2271   0.537543  -0.114948 -0.271526   1.350403
 ...         ...        ...       ...        ...
 11284 -0.867017   0.803453  1.001807   0.725085
 11964 -0.754652   1.067992 -0.423254   0.863130
 5390  -0.749970   0.593818 -0.454777  -0.081700
 860    0.912092  -1.193070  0.643306  -0.622534
 15795  1.001048  -1.422670 -0.544884   0.297998

 [16512 rows x 10 columns],
           MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  \
 20046 -1.152489 -0.289187 -0.499896  -0.156976   -0.029562  0.077681
 3024  -0.705015  0.108104 -0.157477   0.204301    0.123206 -0.037634
 15663 -0.205588  1.856182 -0.586814   0.188231   -0.101972 -0.164679
 20484  0.982710 -0.924851  0.296929  -0.161370    0.246834  0.035990
 9814  -0.076678  0.425936  0.025864  -0.144836   -0.320086 -0.056520
 ...         ...       ...       ...        ...         ...       ...
 15362  0.386535 -1.004309  0.635869  -0.063945   -0.065767 -0.007868
 16623 -0.602214 -0.050812  0.284108   0.337756    0.198266 -0.070310
 18086  2.820927 -0.289187  0.731040  -0.315451    0.140868 -0.026976
 2144  -0.571473  0.584852 -0.056574  -0.239614   -0.175266 -0.046414
 3665  -0.167689 -0.924851 -0.582092  -0.133347    0.216810  0.063477

        Latitude  Longitude  Income^2  Age*Rooms
 20046  0.200449   0.279366 -0.718345  -0.521680
 3024  -0.230283   0.054757 -0.555170   0.005346
 15663  1.015093  -1.432653 -0.295277   0.637535
 20484 -0.632923   0.424114  0.652912  -0.521944
 9814   0.462633  -1.178096 -0.214874   0.409681
 ...         ...        ...       ...        ...
 15362 -1.063655   1.172810  0.119146  -0.439513
 16623 -0.127282  -0.629052 -0.508379   0.238447
 18086  0.785682  -1.237992  3.034627   0.343545
 2144   0.532861  -0.094982 -0.493712   0.451085
 3665  -0.661015   0.598809 -0.272206  -0.942091

 [4128 rows x 10 columns],
 14196    1.030
 8267     3.821
 17445    1.726
 14265    0.934
 2271     0.965
          ...
 11284    2.292
 11964    0.978
 5390     2.221
```

```
860      2.835
15795    3.250
Length: 16512, dtype: float64,
20046    0.47700
3024     0.45800
15663    5.00001
20484    2.18600
9814     2.78000
          ...
15362    2.63300
16623    2.66800
18086    5.00001
2144     0.72300
3665     1.51500
Length: 4128, dtype: float64)
```

In [10]:
```python
xgb_model = xgb.XGBRegressor(objective='reg:squarederror')
xgb_model
```

Out[10]:
```
XGBRegressor

XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=None, n_jobs=None,
```

In [11]:
```python
params = {
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01, 0.05, 0.1],
    'n_estimators': [100, 300, 500],
    'subsample': [0.8, 0.95, 1.0],
    'colsample_bytree': [0.5, 0.75, 1.0]
}
```

In [12]:
```python
grid_search = GridSearchCV(estimator=xgb_model, param_grid=params, cv=5, scoring=
grid_search.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 243 candidates, totalling 1215 fits
```

Out[12]:
```
GridSearchCV
estimator: XGBRegressor
XGBRegressor
```

In [13]:
```
best_xgb = grid_search.best_estimator_
best_xgb
```

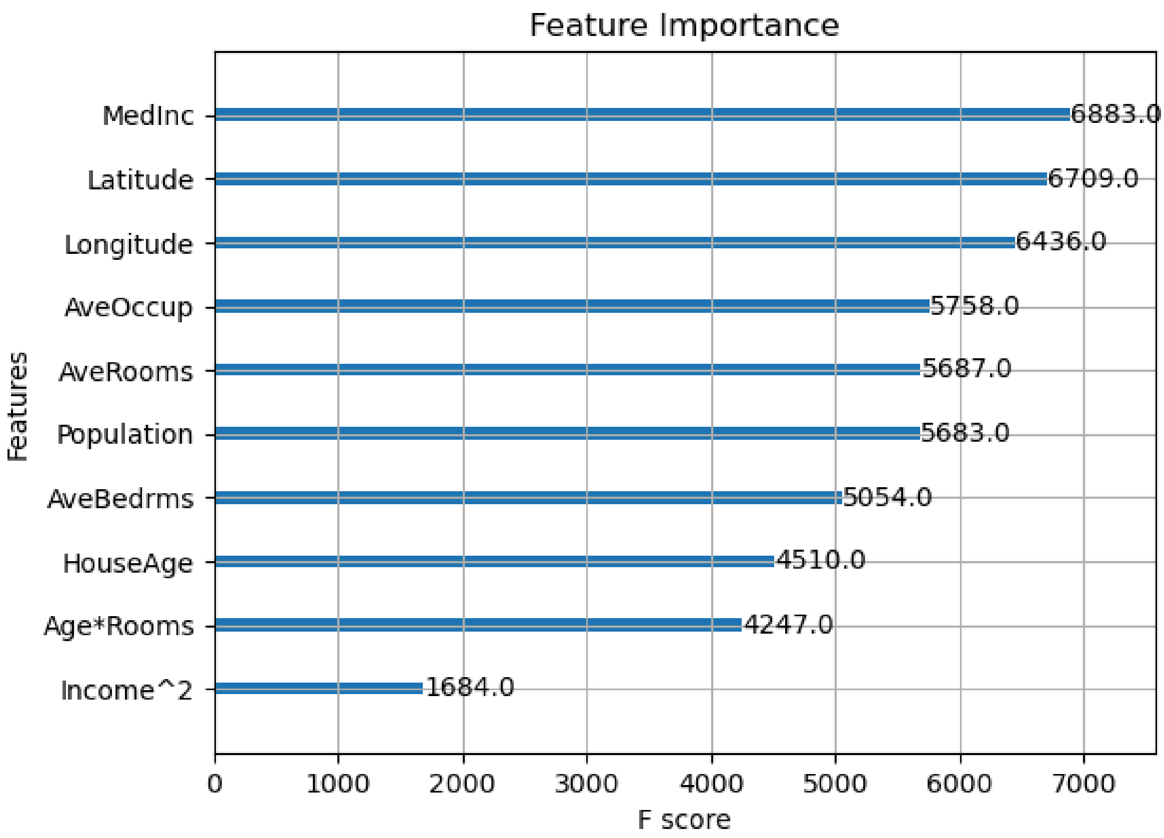
Out[13]:
```
XGBRegressor

XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=0.75, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=0.05, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=7, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=500, n_jobs=None,
```

In [14]:
```
y_pred = best_xgb.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f"The MSE of the optimized XGBoost model is: {mse}")
```

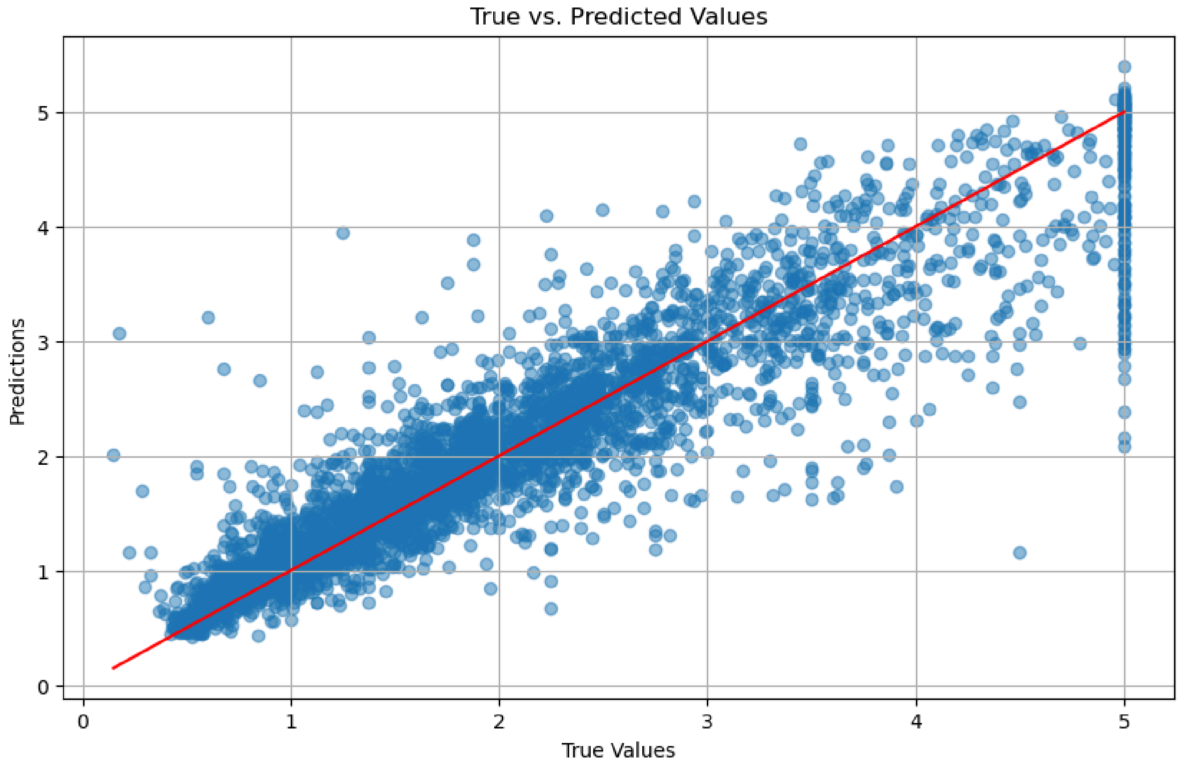
```
The MSE of the optimized XGBoost model is: 0.19794659552807892
```

In [15]:
```
xgb.plot_importance(best_xgb)
plt.title('Feature Importance')
plt.show()
```

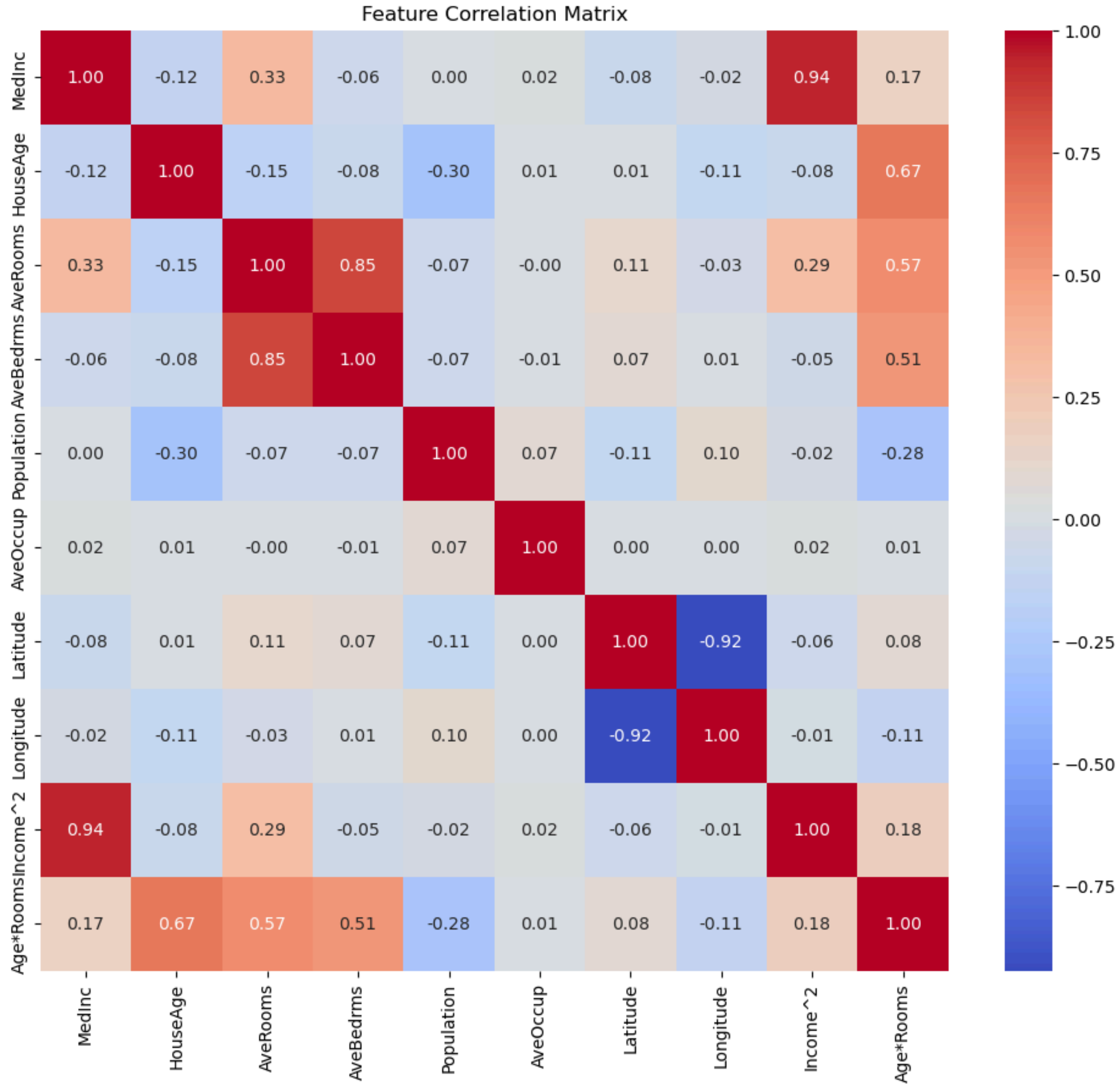


In [16]:

```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.xlabel('True Values')
plt.ylabel('Predictions')
plt.title('True vs. Predicted Values')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red')  # 
plt.grid(True)
plt.show()
```



In [17]:
```python
plt.figure(figsize=(12, 10))
sns.heatmap(X.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Feature Correlation Matrix')
plt.show()
```



In [20]:
```python
from sklearn.inspection import PartialDependenceDisplay
```

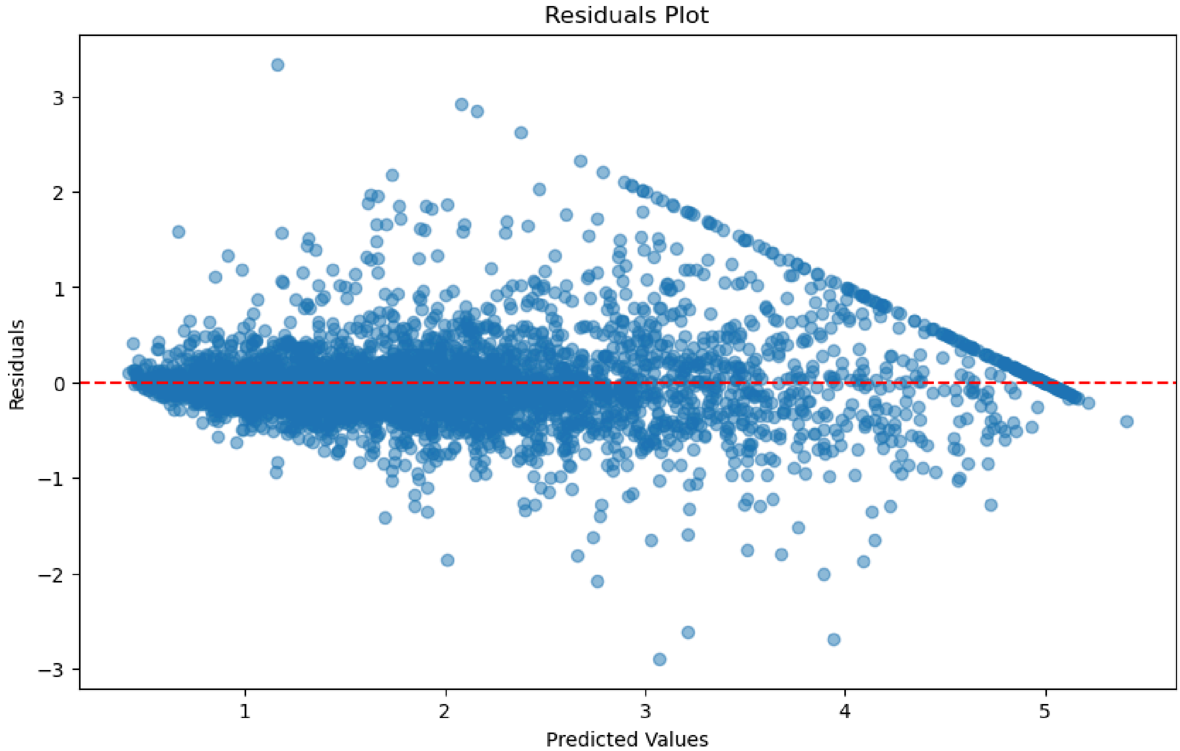
In [21]:
```
!pip install shap
```

```
Collecting shap
  Obtaining dependency information for shap from https://files.pythonhosted.org/packages/40/6c/850cdf7d0c6351ee9d060c0a24237381ae212c125553afa61198eaa06b0b/shap-0.45.1-cp311-cp311-win_amd64.whl.metadata (https://files.pythonhosted.org/packages/40/6c/850cdf7d0c6351ee9d060c0a24237381ae212c125553afa61198eaa06b0b/shap-0.45.1-cp311-cp311-win_amd64.whl.metadata)
  Downloading shap-0.45.1-cp311-cp311-win_amd64.whl.metadata (25 kB)
Requirement already satisfied: numpy in c:\users\aditya kudva\anaconda3\lib\site-packages (from shap) (1.24.3)
Requirement already satisfied: scipy in c:\users\aditya kudva\anaconda3\lib\site-packages (from shap) (1.10.1)
Requirement already satisfied: scikit-learn in c:\users\aditya kudva\anaconda3\lib\site-packages (from shap) (1.3.0)
Requirement already satisfied: pandas in c:\users\aditya kudva\anaconda3\lib\site-packages (from shap) (1.5.3)
Requirement already satisfied: tqdm>=4.27.0 in c:\users\aditya kudva\anaconda3\lib\site-packages (from shap) (4.65.0)
Requirement already satisfied: packaging>20.9 in c:\users\aditya kudva\anaconda3\lib\site-packages (from shap) (23.2)
Collecting slicer==0.0.8 (from shap)
  Obtaining dependency information for slicer==0.0.8 from https://files.pythonhosted.org/packages/63/81/9ef641ff4e12cbcca30e54e72fb0951a2ba195d0cda0ba4100e532d929db/slicer-0.0.8-py3-none-any.whl.metadata (https://files.pythonhosted.org/packages/63/81/9ef641ff4e12cbcca30e54e72fb0951a2ba195d0cda0ba4100e532d929db/slicer-0.0.8-py3-none-any.whl.metadata)
  Downloading slicer-0.0.8-py3-none-any.whl.metadata (4.0 kB)
Requirement already satisfied: numba in c:\users\aditya kudva\anaconda3\lib\site-packages (from shap) (0.57.0)
Requirement already satisfied: cloudpickle in c:\users\aditya kudva\anaconda3\lib\site-packages (from shap) (2.2.1)
Requirement already satisfied: colorama in c:\users\aditya kudva\anaconda3\lib\site-packages (from tqdm>=4.27.0->shap) (0.4.6)
Requirement already satisfied: llvmlite<0.41,>=0.40.0dev0 in c:\users\aditya kudva\anaconda3\lib\site-packages (from numba->shap) (0.40.0)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\aditya kudva\anaconda3\lib\site-packages (from pandas->shap) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\aditya kudva\anaconda3\lib\site-packages (from pandas->shap) (2022.7)
Requirement already satisfied: joblib>=1.1.1 in c:\users\aditya kudva\anaconda3\lib\site-packages (from scikit-learn->shap) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\aditya kudva\anaconda3\lib\site-packages (from scikit-learn->shap) (2.2.0)
Requirement already satisfied: six>=1.5 in c:\users\aditya kudva\anaconda3\lib\site-packages (from python-dateutil>=2.8.1->pandas->shap) (1.16.0)
Downloading shap-0.45.1-cp311-cp311-win_amd64.whl (455 kB)
   ---------------------------------------- 0.0/455.5 kB ? eta -:--:--
   -- ------------------------------------- 30.7/455.5 kB 1.3 MB/s eta 0:00:01
   --- ------------------------------------ 41.0/455.5 kB 487.6 kB/s eta 0:00:01
   --- ------------------------------------ 41.0/455.5 kB 487.6 kB/s eta 0:00:01
   --- ------------------------------------ 41.0/455.5 kB 487.6 kB/s eta 0:00:01
   ----- ---------------------------------- 61.4/455.5 kB 272.3 kB/s eta 0:00:02
   ------- -------------------------------- 92.2/455.5 kB 348.6 kB/s eta 0:00:02
   --------- ------------------------------ 112.6/455.5 kB 327.2 kB/s eta 0:00:02
   ---------- ----------------------------- 122.9/455.5 kB 312.9 kB/s eta 0:00:02
   ----------- ---------------------------- 143.4/455.5 kB 340.5 kB/s eta 0:00:01
   ------------- -------------------------- 163.8/455.5 kB 350.7 kB/s eta 0:00:01
   -------------- ------------------------- 174.1/455.5 kB 337.8 kB/s eta 0:00:01
   ---------------- ----------------------- 194.6/455.5 kB 357.2 kB/s eta 0:00:01
   --------------------- ------------------ 245.8/455.5 kB 396.5 kB/s eta 0:00:01
   ---------------------- ----------------- 256.0/455.5 kB 393.2 kB/s eta 0:00:01
   ---------------------- ----------------- 256.0/455.5 kB 393.2 kB/s eta 0:00:01
   -------------------------- ------------- 307.2/455.5 kB 421.8 kB/s eta 0:00:01
   ----------------------------- ---------- 337.9/455.5 kB 427.8 kB/s eta 0:00:01
```

```
   -------------------------------- ----- 389.1/455.5 kB 466.0 kB/s eta 0:00:01
   ------------------------------------ - 440.3/455.5 kB 500.1 kB/s eta 0:00:01
   -------------------------------------  450.6/455.5 kB 494.2 kB/s eta 0:00:01
   -------------------------------------- 455.5/455.5 kB 466.7 kB/s eta 0:00:00
Downloading slicer-0.0.8-py3-none-any.whl (15 kB)
Installing collected packages: slicer, shap
Successfully installed shap-0.45.1 slicer-0.0.8
```

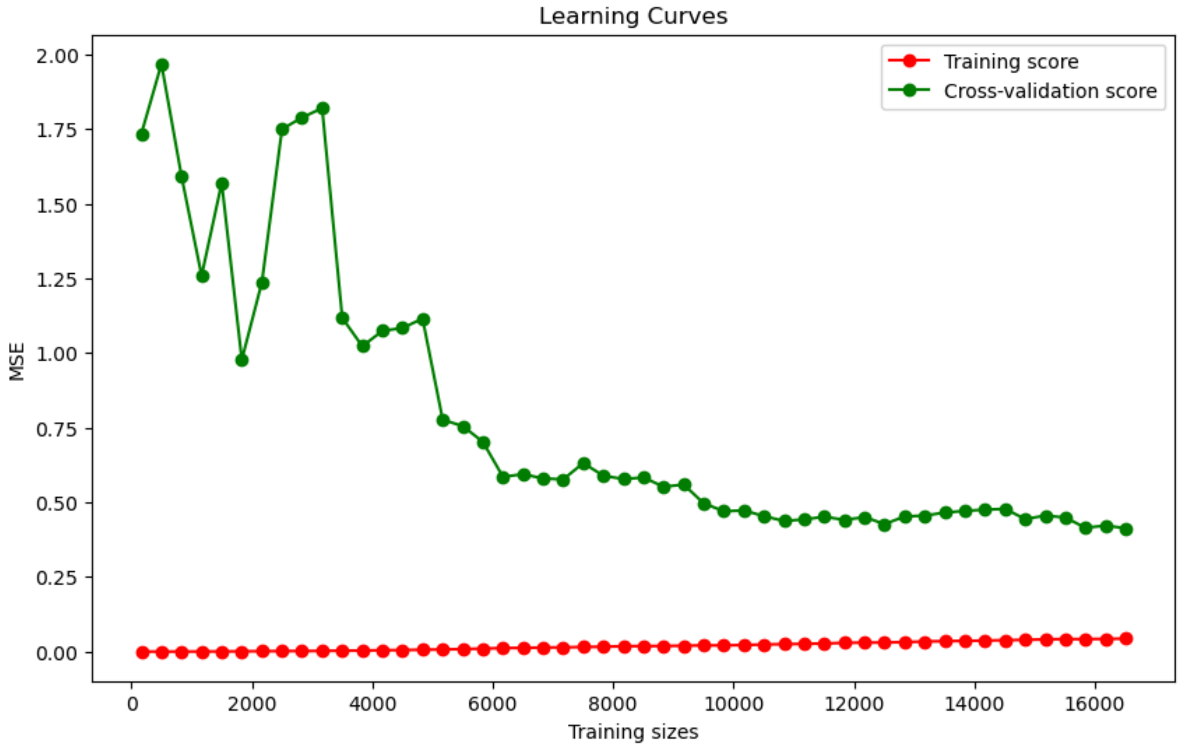
In [29]:
```python
import shap
from sklearn.model_selection import train_test_split, GridSearchCV, learning_curv
```

In [27]:
```python
residuals = y_test - y_pred
plt.figure(figsize=(10, 6))
plt.scatter(y_pred, residuals, alpha=0.5)
plt.axhline(0, color='red', linestyle='--')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.title('Residuals Plot')
plt.show()
```



In [30]:
```python
train_sizes, train_scores, test_scores = learning_curve(best_xgb, X, y, cv=5, sco
train_mean = -np.mean(train_scores, axis=1)
test_mean = -np.mean(test_scores, axis=1)
```

In [31]:
```
plt.figure(figsize=(10, 6))
plt.plot(train_sizes, train_mean, 'o-', color="r", label="Training score")
plt.plot(train_sizes, test_mean, 'o-', color="g", label="Cross-validation score")
plt.xlabel('Training sizes')
plt.ylabel('MSE')
plt.title('Learning Curves')
plt.legend(loc="best")
plt.show()
```



In [32]:
```
explainer = shap.Explainer(best_xgb)
shap_values = explainer(X_test)
```

In [33]:
```
shap.summary_plot(shap_values, X_test, plot_type="bar")
plt.show()
```