This Boston Housing dataset model has practical applications in real estate:

- Automated property valuation for agents, banks, and appraisers
- Investment decision support for real estate investors and developers
- Market analysis to identify price trends and key value drivers
- Risk assessment for mortgage lending and insurance
- Integration into real estate websites and apps for instant estimates
- The model offers data-driven, scalable property assessments, reducing bias and saving time.

In [3]:
```
import pandas as pd
import torch
boston_df = pd.read_csv('/Users/vbaderdi/Downloads/Boston.csv')
dataset_df = pd.read_csv('/Users/vbaderdi/Downloads/dataset.csv')
```

In [4]:
```
boston_df.isna().sum()
```

Out[4]:
```
Unnamed: 0    0
crim          0
zn            0
indus         0
chas          0
nox           0
rm            0
age           0
dis           0
rad           0
tax           0
ptratio       0
black         0
lstat         0
medv          0
dtype: int64
```

In [18]:
```
boston_df.head()
```

Out[18]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

In [5]:
```
y = boston_df.iloc[:,-1]
x = boston_df.iloc[:,:-1]
```

In [6]:
```
from sklearn.model_selection import train_test_split
```

In [7]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.25,shuffle=True)
```

In [8]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.fit_transform(x_test)
```

In [9]:
```
# Define the model

import torch
import torch.nn as nn
class RegressionNet(nn.Module):  # 'Module' not 'module'
    def __init__(self, input_dim=14):
        super(RegressionNet, self).__init__()
        # Layer 1
        self.layer_1 = nn.Linear(14, 28)
        self.bn1 = nn.BatchNorm1d(28)

        # Layer 2
        self.layer_2 = nn.Linear(28, 14)
        self.bn2 = nn.BatchNorm1d(14)

        # Layer 3
        self.layer_3 = nn.Linear(14, 1)

        self.relu = nn.ReLU()
        self.dropout = nn.Dropout(0.2)
```

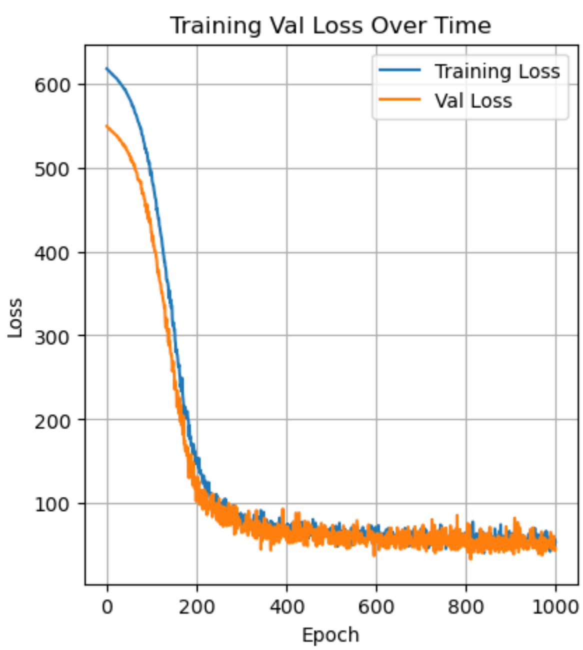
```
    def forward(self,x):
        x = self.dropout(self.relu(self.layer_1(x)))
        x = self.dropout(self.relu(self.layer_2(x)))
        x = self.layer_3(x)
        return x
```

In [10]:
```
# Data prep for training

x_train_scaled = torch.FloatTensor(x_train_scaled)
x_test_scaled = torch.FloatTensor(x_test_scaled)

y_train = torch.FloatTensor(y_train.tolist()).reshape(-1,1)
y_test = torch.FloatTensor(y_test.tolist()).reshape(-1,1)
```

In [11]:
```
# Model training

model = RegressionNet()
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr = 0.001)
num_epoch = 1000
train_loss_plot = []
loss_test = []
for epoch in range(num_epoch):
    model.train()
    outputs = model(x_train_scaled)
    loss = criterion(outputs,y_train)

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    # Test accuracy
    train_loss_plot.append(loss.item())
    loss_test.append(criterion(model(x_test_scaled),y_test).item())
```

In [12]:
```
# Model evaluate

import matplotlib.pyplot as plt
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(train_loss_plot, label='Training Loss')
plt.plot(loss_test, label='Val Loss')
plt.title('Training Val Loss Over Time')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.grid(True)
plt.legend(loc='upper right')  # Specify location
```

Out[12]: `<matplotlib.legend.Legend at 0x19c86ac50>`



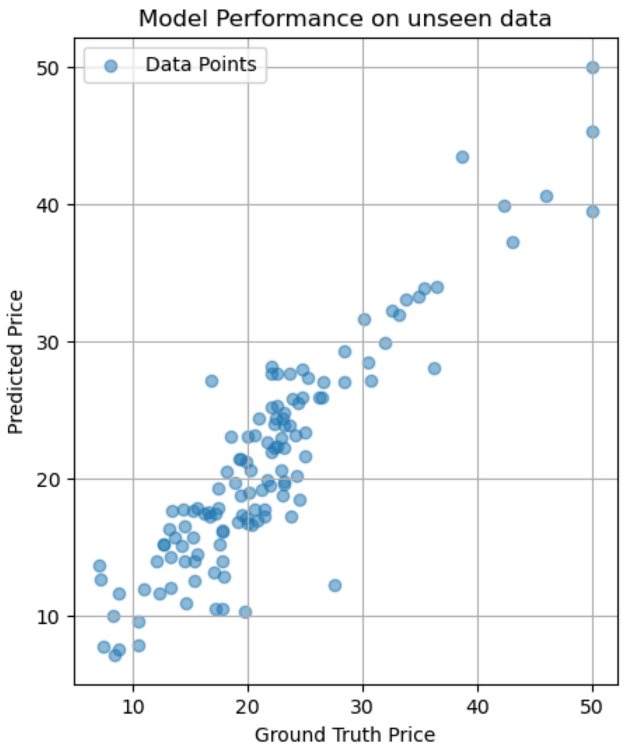
In [13]:
```
# Example of how to evaluate on unseen test data
model.eval()
with torch.no_grad():
    test_predictions = model(x_test_scaled)
```

```python
    mse = criterion(test_predictions, y_test)
    print(f'Test MSE: {mse.item():.4f}')
```

```
Test MSE: 12.8748
```

In [17]:

```python
# Example of how to scatter plots of prediction vs ground truth price from unseen test data
plt.figure(figsize=(5, 6))
plt.scatter(y_test.squeeze().tolist(),test_predictions, alpha=0.5, label='Data Points')
plt.xlabel('Ground Truth Price')
plt.ylabel('Predicted Price')
plt.title('Model Performance on unseen data')
plt.legend()
plt.grid(True)
plt.show()
```



In [ ]: