Comparison of Two Regressions in Scikit

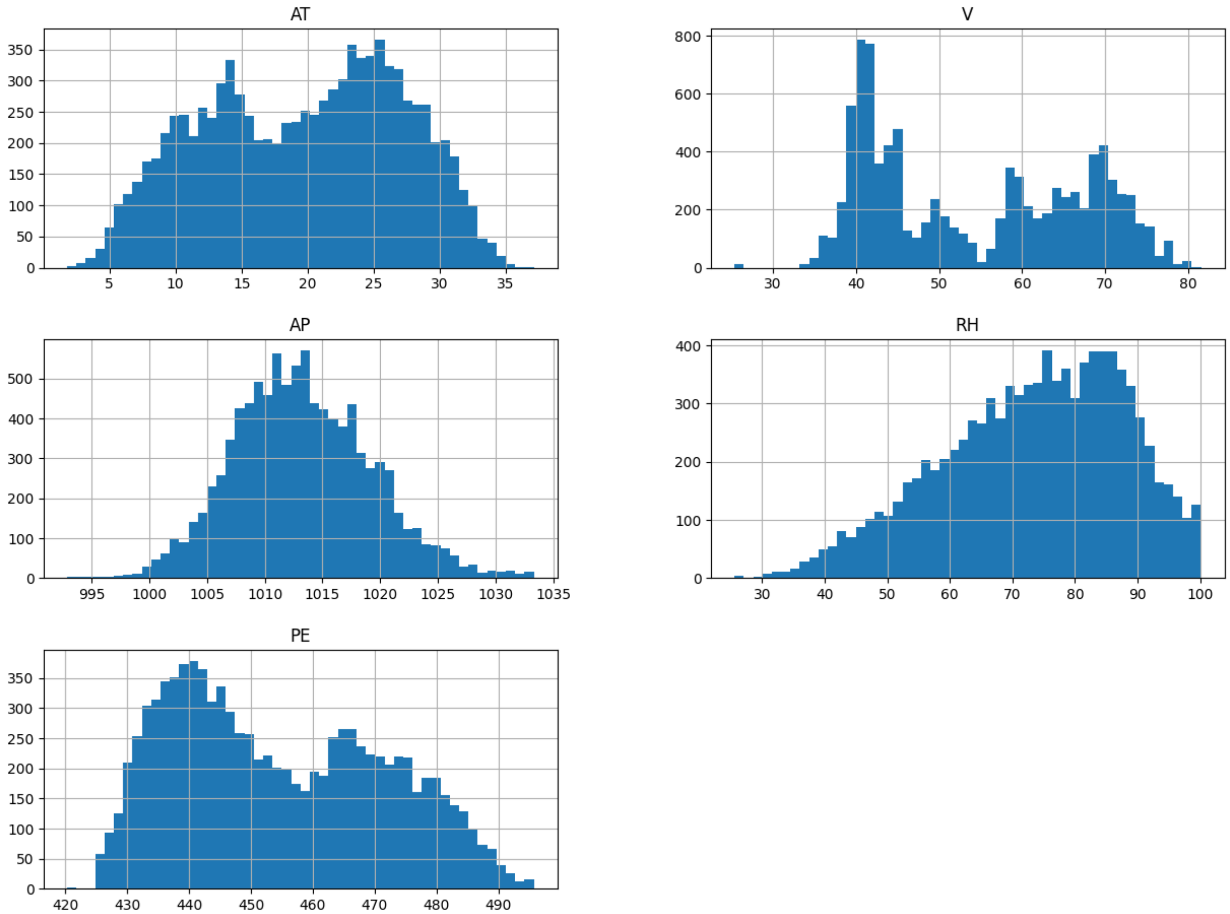
Imports

In [ ]:
```
import pandas as pd
import os
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import StratifiedShuffleSplit
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.experimental import enable_halving_search_cv
from sklearn.model_selection import HalvingGridSearchCV
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import uniform as sp_randFloat
from scipy.stats import randint as sp_randInt
```

Loading the Power Plant Data

In [ ]:
```
def load_pp_data():
    csv_path = r"C:\Users\18123\OneDrive\Documents\IU Bloomington\Machine-Learning-Pro
    return pd.read_csv(csv_path)

pp = load_pp_data()
print(pp.describe())

pp.hist(bins=50, figsize=(15,11))
plt.show()
```

```
                AT            V           AP           RH           PE
count  9568.000000  9568.000000  9568.000000  9568.000000  9568.000000
mean     19.651231    54.305804  1013.259078    73.308978   454.365009
std       7.452473    12.707893     5.938784    14.600269    17.066995
min       1.810000    25.360000   992.890000    25.560000   420.260000
25%      13.510000    41.740000  1009.100000    63.327500   439.750000
50%      20.345000    52.080000  1012.940000    74.975000   451.550000
75%      25.720000    66.540000  1017.260000    84.830000   468.430000
max      37.110000    81.560000  1033.300000   100.160000   495.760000
```

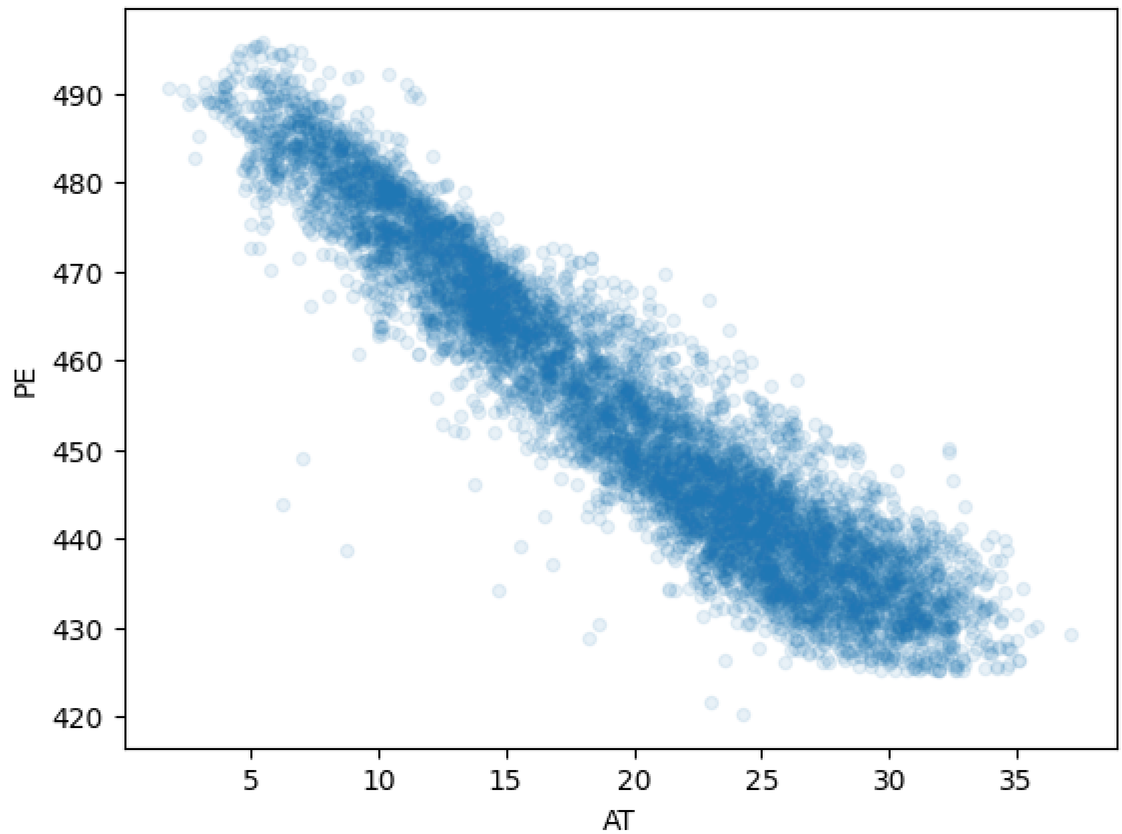


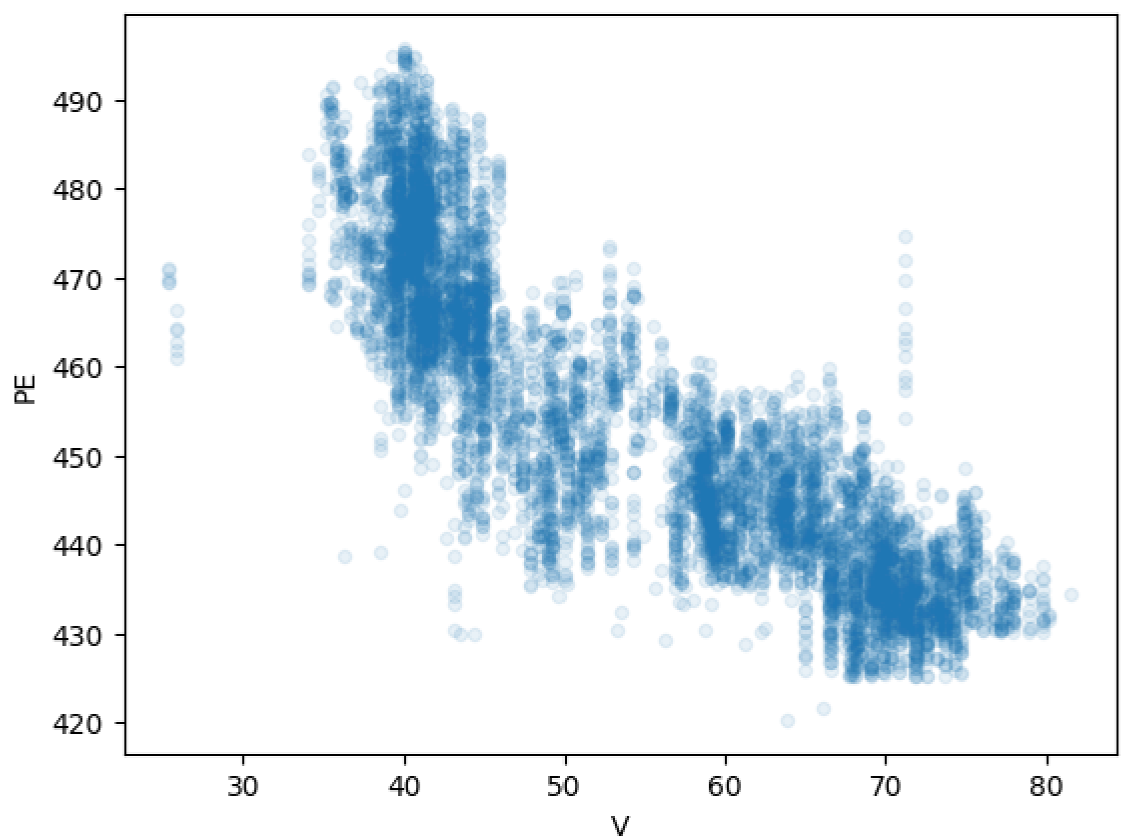
Preprocessing the Data

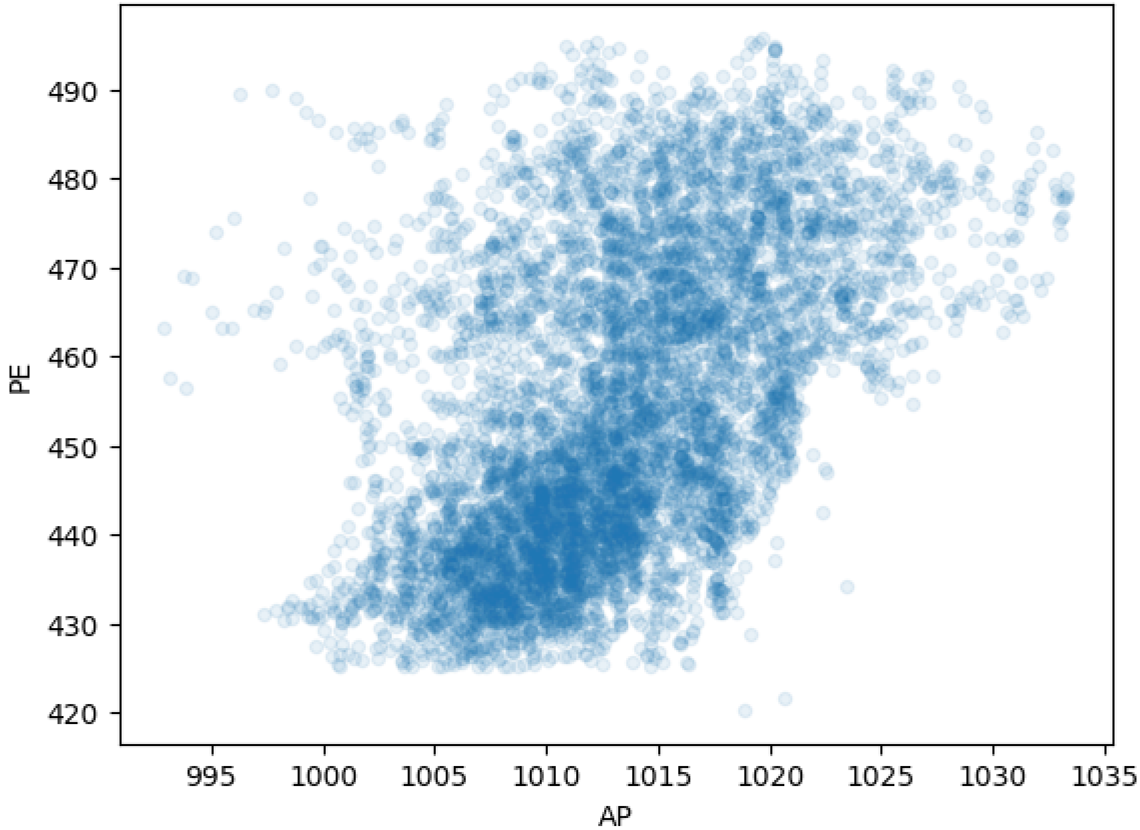
In [ ]:

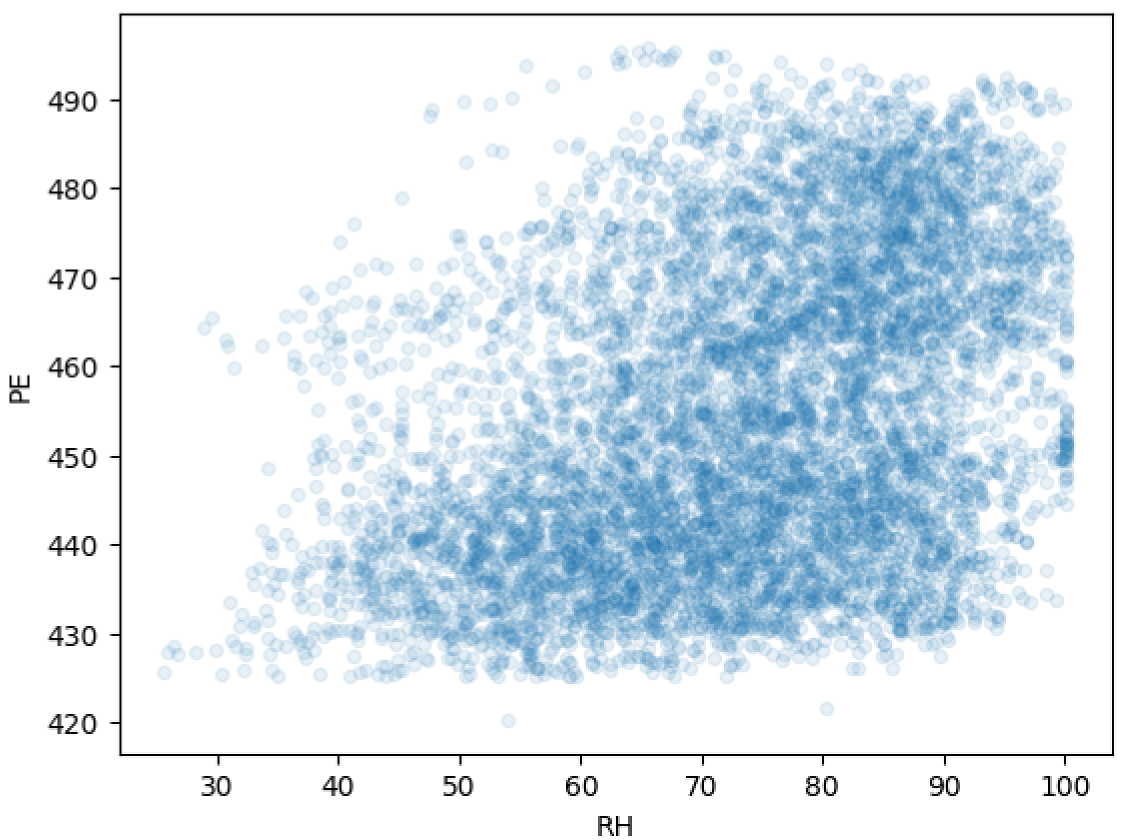
```
pp["AT_cat"] = pd.cut(pp["AT"],bins=[0.,10.,20.,30.,np.inf],labels=[1,2,3,4])

split = StratifiedShuffleSplit(n_splits=1,test_size=0.2,random_state=42)
for train_index, test_index in split.split(pp,pp["AT_cat"]):
    train_set = pp.loc[train_index]
    test_set = pp.loc[test_index]

for set_ in(train_set,test_set):
    set_.drop("AT_cat",axis=1,inplace=True)

pptrain = train_set.copy()
pptest = test_set.copy()

pptrain_attrib = pptrain.drop("PE",axis=1)
pptrain_labels = pptrain["PE"].copy()

pptest_attrib = pptest.drop("PE",axis=1)
pptest_labels = pptest["PE"].copy()

scaler = StandardScaler()
scaler.fit_transform(pptrain_attrib)
scaler.transform(pptest_attrib)

pptrain.plot(kind="scatter",x="AT",y="PE",alpha=0.1)
pptrain.plot(kind="scatter",x="V",y="PE",alpha=0.1)
pptrain.plot(kind="scatter",x="AP",y="PE",alpha=0.1)
```

```
pptrain.plot(kind="scatter",x="RH",y="PE",alpha=0.1)
plt.show()
```









Fitting a Linear Regression with Default Hyperparameters and Performing Cross-Validation

In [ ]:
```
lin_reg = LinearRegression()
lin_reg.fit(pptrain_attrib,pptrain_labels)
```

```
print(lin_reg.get_params())

lin_scores = cross_val_score(lin_reg, pptrain_attrib, pptrain_labels, scoring="neg_mea
lin_rmse_scores = np.sqrt(-lin_scores)

def display_scores(scores):
    print(f"Mean of RMSE: {scores.mean()} +- {scores.std()}")
    print("RMSE:", scores)

display_scores(lin_rmse_scores)
```

```
{'copy_X': True, 'fit_intercept': True, 'n_jobs': None, 'normalize': 'deprecated', 'p
ositive': False}
Mean of RMSE: 4.57453920222217 +- 0.18215801145507707
RMSE: [4.6930045  4.39498373 4.17323854 4.81310111 4.7468393  4.42177793
 4.57160279 4.63866768 4.65619459 4.63598184]
```

Searching Hyperparameters and Performing Cross-Validation to Yield Optimal Linear Regression Model

In [ ]:
```
lin_param_grid = [{'copy_X': [True], 'fit_intercept': [True, False], 'n_jobs': [None],
lin_grid_search = GridSearchCV(lin_reg, lin_param_grid, cv=10, scoring='neg_mean_squar
lin_grid_search.fit(pptrain_attrib,pptrain_labels)
print(lin_grid_search.best_params_)
```

```
{'copy_X': True, 'fit_intercept': True, 'n_jobs': None, 'normalize': 'deprecated', 'p
ositive': False}
```

Fitting a Decision Tree with Default Hyperparameters and Performing Cross-Validation

In [ ]:
```
tree_reg = DecisionTreeRegressor(random_state=10)
tree_reg.fit(pptrain_attrib,pptrain_labels)

print(tree_reg.get_params())

tree_scores = cross_val_score(tree_reg, pptrain_attrib, pptrain_labels, scoring="neg_m
tree_rmse_scores = np.sqrt(-tree_scores)

display_scores(tree_rmse_scores)
```

```
{'ccp_alpha': 0.0, 'criterion': 'squared_error', 'max_depth': None, 'max_features': N
one, 'max_leaf_nodes': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'mi
n_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'random_state': 10, 'splitter':
'best'}
Mean of RMSE: 4.527654138264067 +- 0.20062538151074863
RMSE: [4.54979376 4.65365104 4.20049399 4.73400805 4.82929986 4.38398883
 4.42459583 4.584338   4.68336236 4.23300967]
```

Searching Hyperparameters and Performing Cross-Validation to Yield Optimal Decision Tree Model

In [ ]:
```
tree_param_grid = [{'ccp_alpha': [0.0,0.00001], 'criterion': ['squared_error','friedma
tree_grid_search = HalvingGridSearchCV(tree_reg, tree_param_grid, cv=10, factor=3, max
tree_grid_search.fit(pptrain_attrib,pptrain_labels)
print(tree_grid_search.best_params_)
```

```
{'ccp_alpha': 0.0, 'criterion': 'absolute_error', 'max_depth': None, 'max_features':
'log2', 'max_leaf_nodes': 10, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'm
in_samples_split': 4, 'min_weight_fraction_leaf': 0.0, 'random_state': 10, 'splitte
r': 'best'}
```

Measuring Regression Performance

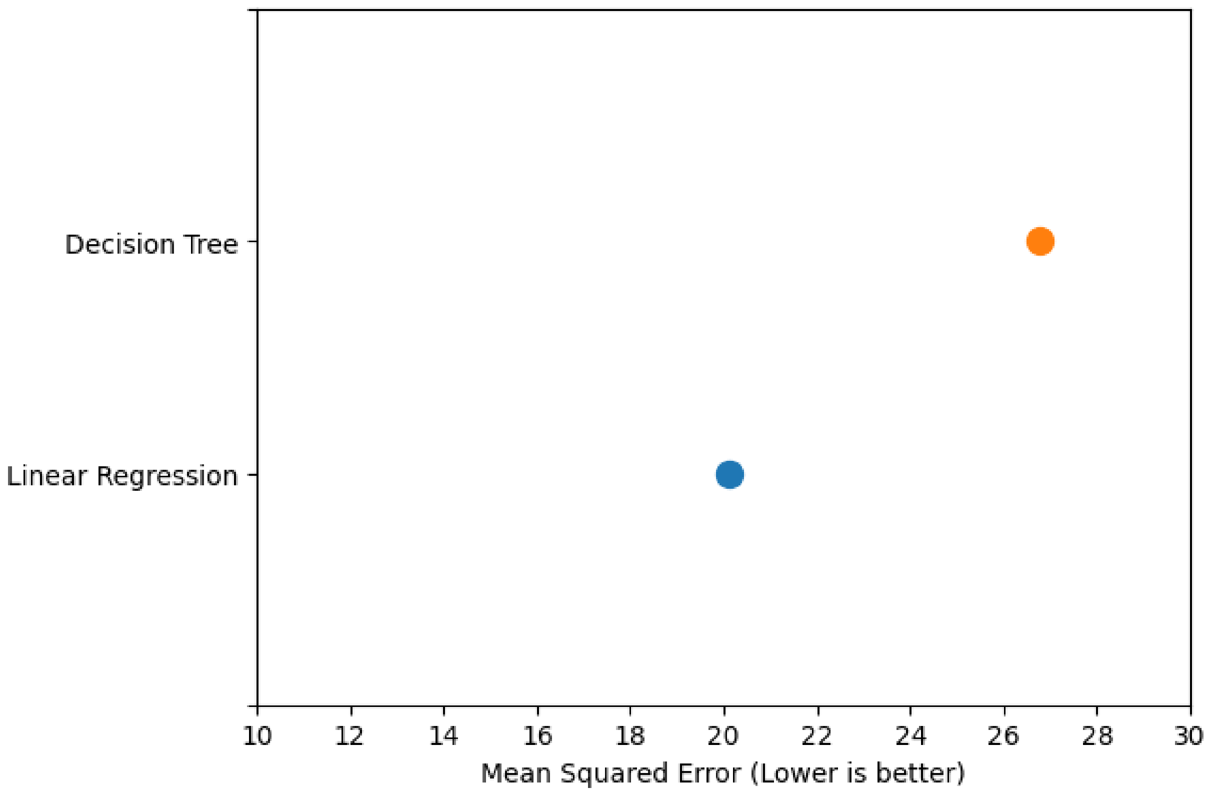
In [ ]:
```
final_lin = lin_grid_search.best_estimator_
final_tree = tree_grid_search.best_estimator_

final_lin_predictions = final_lin.predict(pptest_attrib)
final_tree_predictions = final_tree.predict(pptest_attrib)

final_lin_mse = mean_squared_error(pptest_labels,final_lin_predictions)
final_lin_rmse = np.sqrt(final_lin_mse)
final_lin_r2 = r2_score(pptest_labels,final_lin_predictions)
final_lin_mae = mean_absolute_error(pptest_labels,final_lin_predictions)

final_tree_mse = mean_squared_error(pptest_labels,final_tree_predictions)
final_tree_rmse = np.sqrt(final_tree_mse)
final_tree_r2 = r2_score(pptest_labels,final_tree_predictions)
final_tree_mae = mean_absolute_error(pptest_labels,final_tree_predictions)

print(f"Mean Squared Error-\nLinear Model: {final_lin_mse}\nDecision Tree: {final_tree
print(f"Root Mean Squared Error-\nLinear Model: {final_lin_rmse}\nDecision Tree: {fina
print(f"R-Squared Score Score-\nLinear Model: {final_lin_r2}\nDecision Tree: {final_tr
print(f"Mean Absolute Error-\nLinear Model: {final_lin_mae}\nDecision Tree: {final_tre
```

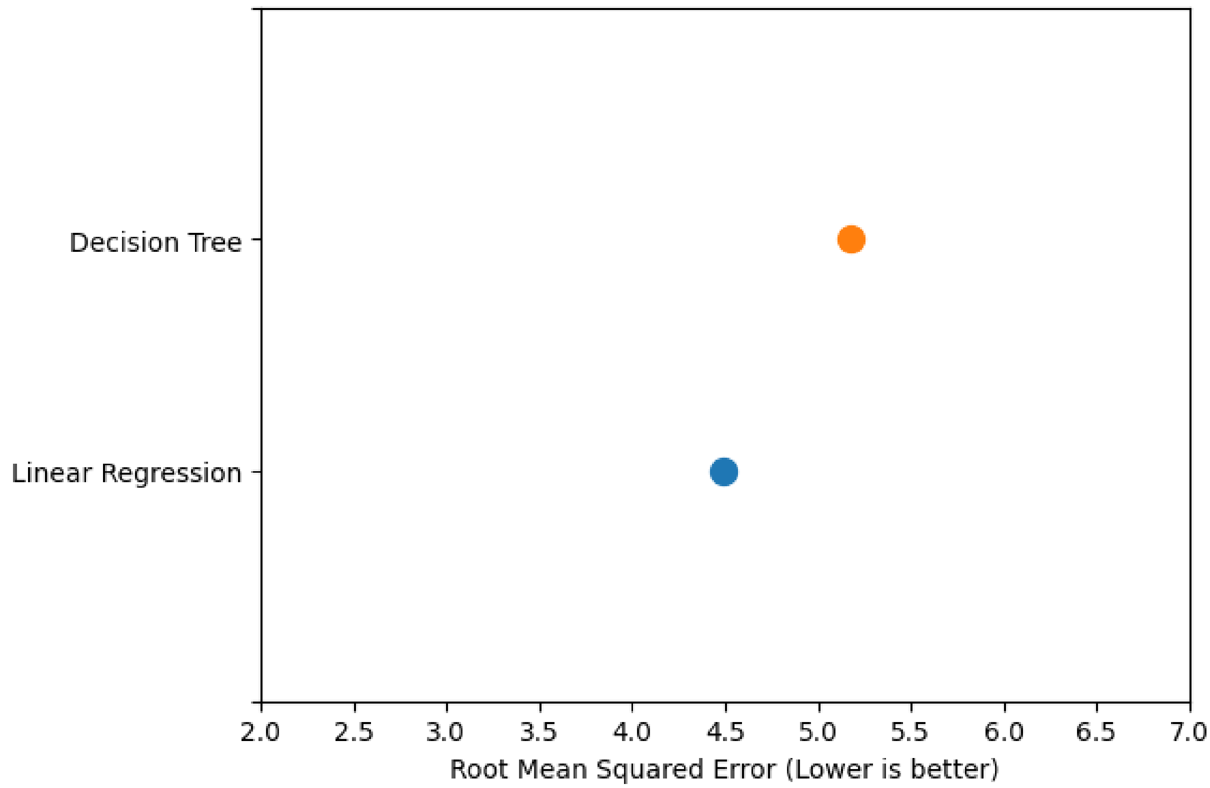
```
Mean Squared Error-
Linear Model: 20.12577739913872
Decision Tree: 26.788148994252868

Root Mean Squared Error-
Linear Model: 4.486176255915356
Decision Tree: 5.175726904914214

R-Squared Score Score-
Linear Model: 0.9318955457576411
Decision Tree: 0.9093504697366541

Mean Absolute Error-
Linear Model: 3.5608570833389614
Decision Tree: 3.979490595611285
```

Vizualizing Regression Performance

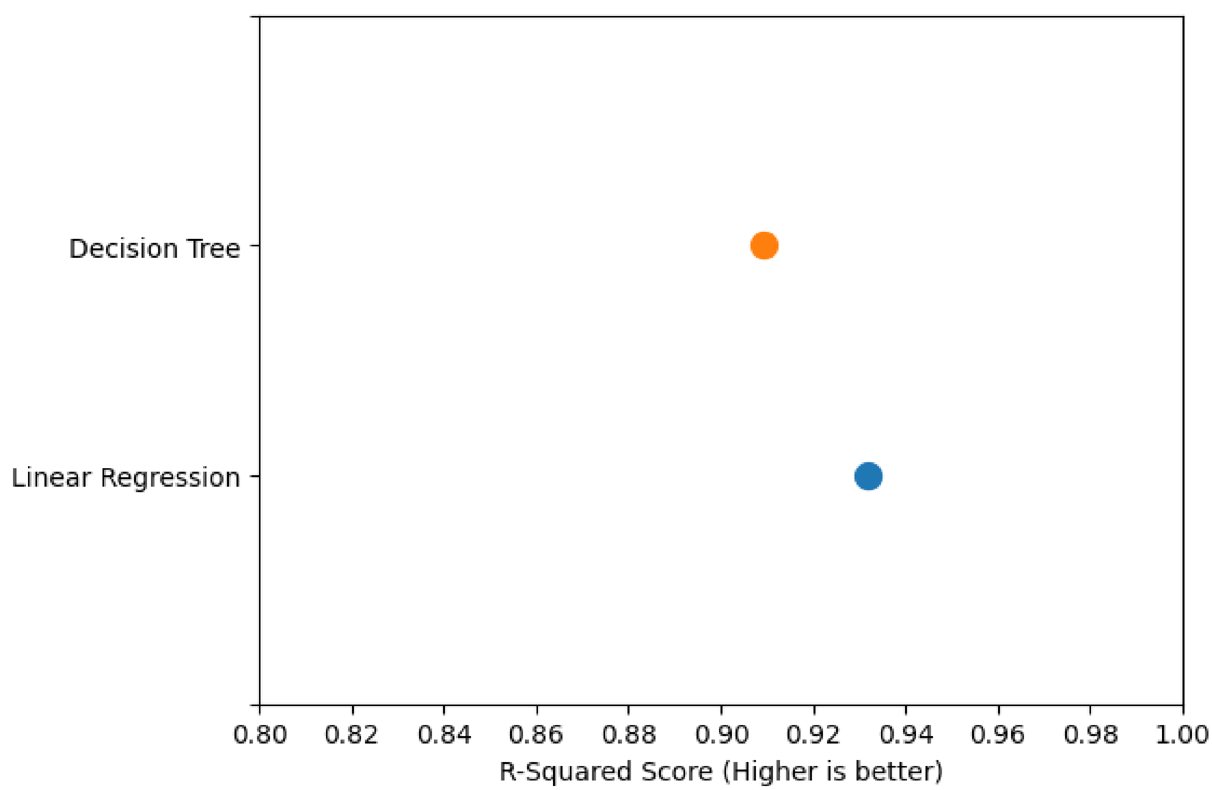
In [ ]:
```
plt.scatter(final_lin_mse,1,s=100)
plt.scatter(final_tree_mse,2,s=100)
plt.xticks([10,12,14,16,18,20,22,24,26,28,30])
plt.yticks([0,1,2,3],labels=['','Linear Regression','Decision Tree',''])
plt.xlabel('Mean Squared Error (Lower is better)')
plt.show()

plt.scatter(final_lin_rmse,1,s=100)
plt.scatter(final_tree_rmse,2,s=100)
plt.xticks([2,2.5,3,3.5,4,4.5,5,5.5,6,6.5,7])
```

```
plt.yticks([0,1,2,3],labels=['','Linear Regression','Decision Tree',''])
plt.xlabel('Root Mean Squared Error (Lower is better)')
plt.show()

plt.scatter(final_lin_r2,1,s=100)
plt.scatter(final_tree_r2,2,s=100)
plt.xticks([0.8,0.82,0.84,0.86,0.88,0.9,0.92,0.94,0.96,0.98,1])
plt.yticks([0,1,2,3],labels=['','Linear Regression','Decision Tree',''])
plt.xlabel('R-Squared Score (Higher is better)')
plt.show()

plt.scatter(final_lin_mae,1,s=100)
plt.scatter(final_tree_mae,2,s=100)
plt.xticks([3,3.25,3.5,3.75,4,4.25,4.5,4.75,5,5.25,5.5,5.75,6])
plt.yticks([0,1,2,3],labels=['','Linear Regression','Decision Tree',''])
plt.xlabel('Mean Absolute Error (Lower is better)')
plt.show()
```

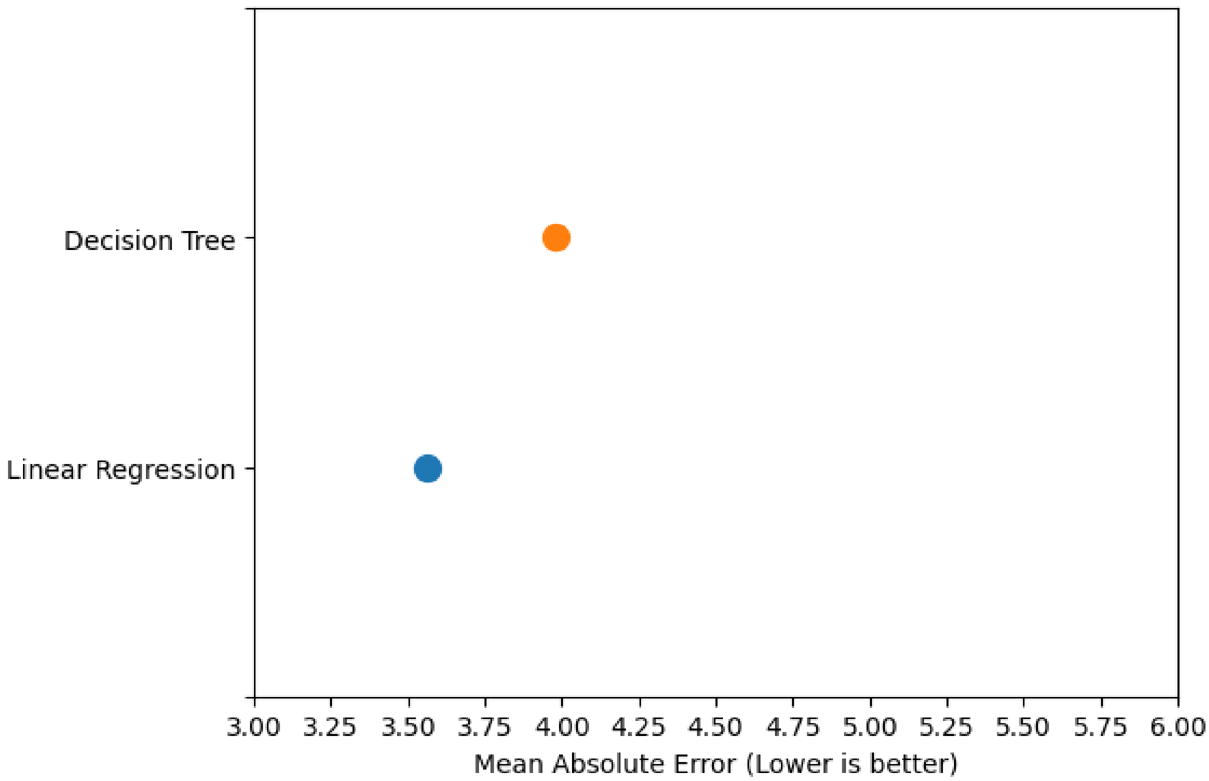




Decision Tree
Linear Regression
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
Root Mean Squared Error (Lower is better)



Decision Tree
Linear Regression
0.80
0.82
0.84
0.86
0.88
0.90
0.92
0.94
0.96
0.98
1.00
R-Squared Score (Higher is better)



Decision Tree
Linear Regression
3.00
3.25
3.50
3.75
4.00
4.25
4.50
4.75
5.00
5.25
5.50
5.75
6.00
Mean Absolute Error (Lower is better)