In [86]:
```
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import gc
import datetime
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.model_selection import learning_curve
from sklearn.metrics import SCORERS
from sklearn.tree import export_graphviz
import IPython, graphviz, re, math
warnings.simplefilter('ignore')
matplotlib.rcParams['figure.dpi'] = 100
sns.set()
```

In [87]:
```
# Extract a minibatch from the original training dataset
def TrTeSplit(X_tr, y_tr, size):
    X_trspl, X_tespl, y_trspl, y_tespl = train_test_split(X_tr, y_tr, test_size=size, random_state=0)
    X_trspl = X_trspl.reset_index(drop = True)
    X_tespl = X_tespl.reset_index(drop = True)
    y_trspl = y_trspl.reset_index(drop = True)
    y_tespl = y_tespl.reset_index(drop = True)

    return X_trspl, X_tespl, y_trspl, y_tespl

# Parameter Tunning using cross validation
def model_param_select(X, y, nfolds):
    max_depth = [20, 21, 22]
    min_samples_split = [2, 3]
    min_samples_leaf = [4, 5, 6]
    #the range is being reduced since after several runs
    #I have removed some redundant range value that will yield longer run time
    param_grid = {'max_depth': max_depth, 'min_samples_split': min_samples_split, 'min_samples_leaf': min_samples_leaf}
    grid_search = GridSearchCV(DecisionTreeRegressor(random_state = 0), param_grid, cv=nfolds)
    grid_search.fit(X, y)
    grid_search.best_params_
    return grid_search.best_params_

def RMSE(y_test, y_pred):
    rss=((y_test-y_pred)**2).sum()
    mse=np.mean((y_test-y_pred)**2)
    rmse = np.sqrt(np.mean((y_test-y_pred)**2))
    return rmse

def DataPre(new_train):
    y_tr = new_train['meter_reading']
    X_tr = new_train.drop(columns = ['meter_reading', 'timestamp', 'Unnamed: 0'])
    X_trspl, X_tespl, y_trspl, y_tespl = TrTeSplit(X_tr, y_tr, 0.33)
    del X_tr, y_tr
    return X_trspl, X_tespl, y_trspl, y_tespl

def LearningVisual(model, X_trspl, y_trspl, X_tespl, y_tespl, percentSize):
    # Visualize learning curves
    plt.figure()
    train_sizes, train_scores, test_scores = \
    learning_curve(model, X_trspl, y_trspl, train_sizes=np.linspace(0.1, 1, percentSize),
                   scoring="neg_root_mean_squared_error", cv=10)
    train_sizes, val_train_scores, val_test_scores = \
    learning_curve(model, X_tespl, y_tespl, train_sizes=np.linspace(0.1, 1, percentSize),
                   scoring="neg_root_mean_squared_error", cv=10)
    plt.plot(train_sizes, -test_scores.mean(1), 'o-', color="r",label="train set")
    plt.plot(train_sizes, -val_test_scores.mean(1), 'o-', color ='b', label="validation set")
    plt.xlabel("Dataset size")
    plt.ylabel("Root Mean Squared Error")
    plt.title('Learning curves')
    plt.legend(loc="best")

    plt.show()
```

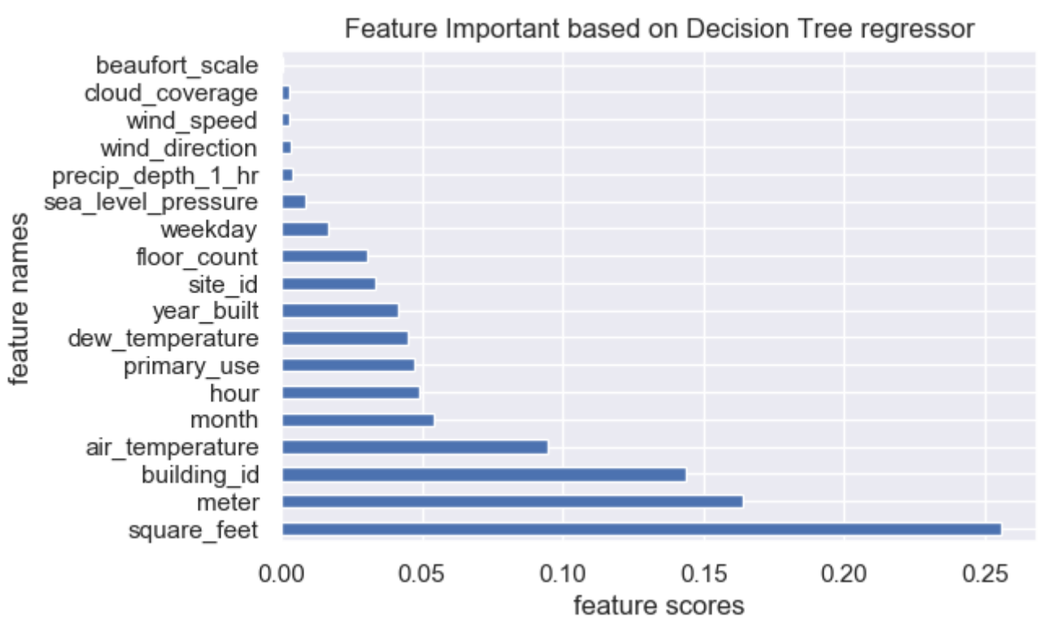
In [78]:
```
# Load Data
new_train = pd.read_csv('sample_train_tiny.csv')
X_trspl, X_tespl, y_trspl, y_tespl = DataPre(new_train)
del new_train
```

In [31]:
```
# Parameter tuning using Cross Validation
print(model_param_select(X_trspl, y_trspl, 10))
```

```
{'max_depth': 20, 'min_samples_leaf': 6, 'min_samples_split': 2}
```

In [96]:
```
rs = 1; md = 20; mss = 6; msl = 2;
#Implement Tree model with the choosen parameter
model = DecisionTreeRegressor(random_state = rs, max_depth = md, min_samples_split = mss, min_samples_leaf = msl)
model.fit(X_trspl, y_trspl)
y_pred = model.predict(X_tespl)

rmse = RMSE(y_tespl, y_pred)
print(rmse)

feat_importances = pd.Series(model.feature_importances_, index=X_trspl.columns)
feat_importances.nlargest(20).plot(kind='barh')
plt.title('Feature Important based on Decision Tree regressor')
plt.xlabel('feature scores')
plt.ylabel('feature names')
plt.show()
```

```
263.0140043934035
```

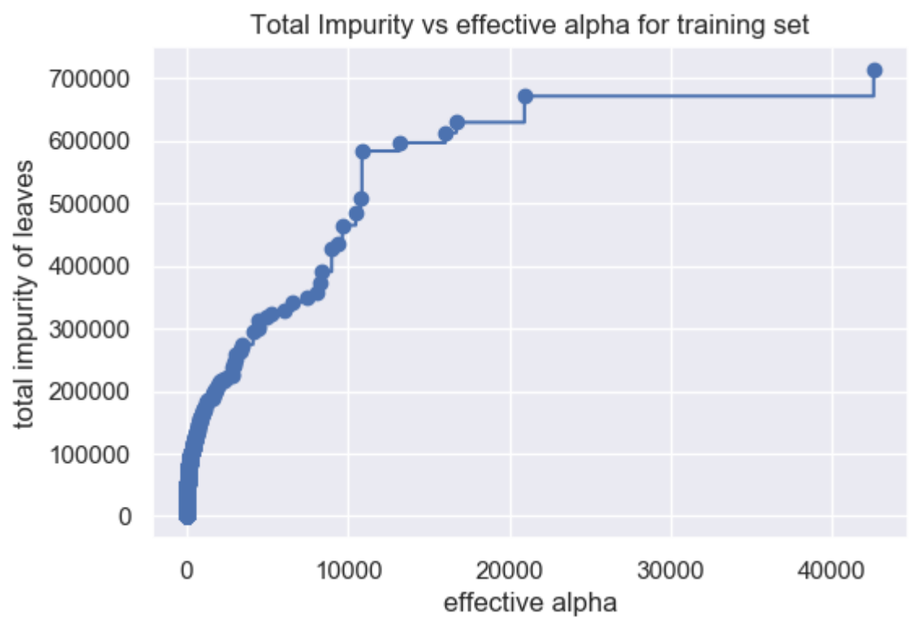


In [67]:
```
# Total impurity of leaves vs effective alphas of pruned tree
path = model.cost_complexity_pruning_path(X_trspl, y_trspl)
ccp_alphas, impurities = path.ccp_alphas, path.impurities
# new_ccp_alphas = []
# new_impurities = []
# for i in np.linspace(0,ccp_alphas.size - 1,100, dtype = int):
#     new_ccp_alphas.append(ccp_alphas[i])
#     new_impurities.append(impurities[i])
fig, ax = plt.subplots()
ax.plot(ccp_alphas[:-1], impurities[:-1], marker='o', drawstyle="steps-post")
ax.set_xlabel("effective alpha")
ax.set_ylabel("total impurity of leaves")
ax.set_title("Total Impurity vs effective alpha for training set")
```

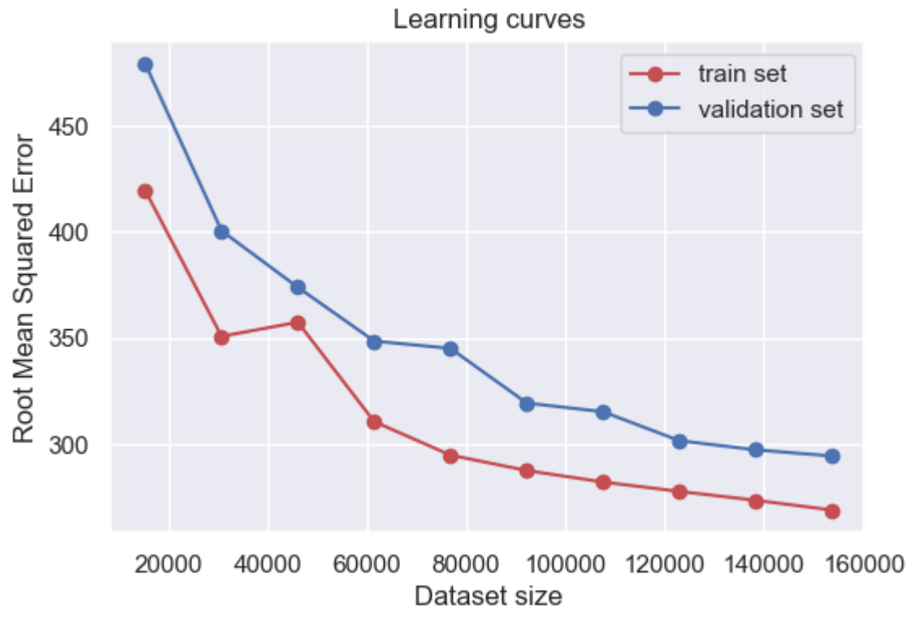
```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-67-099e6312054d> in <module>
      5 new_impurities = []
      6 for i in np.linspace(0,ccp_alphas.size - 1,100, dtype = int):
----> 7     new_ccp_alphas.append(ccp_alpha[i])
      8     new_impurities.append(impurities[i])
      9 fig, ax = plt.subplots()

IndexError: invalid index to scalar variable.
```

In [75]:
```
fig, ax = plt.subplots()
ax.plot(ccp_alphas[:-1], impurities[:-1], marker='o', drawstyle="steps-post")
ax.set_xlabel("effective alpha")
ax.set_ylabel("total impurity of leaves")
ax.set_title("Total Impurity vs effective alpha for training set")
```

Out[75]: Text(0.5, 1.0, 'Total Impurity vs effective alpha for training set')



In [51]:
```
LearningVisual(model, X_trspl, y_trspl, X_tespl, y_tespl, 10)
```



In [9]:
```
X_train = pd.read_csv(r'Dataset/new_train.csv')
```

In [10]:
```
y_train = X_train['meter_reading']
X_train = X_train.drop(columns = ['meter_reading', 'timestamp'])
X_test = X_test.drop(columns = ['meter_reading', 'timestamp'])
```

In [13]:
```
X_trspl, X_tespl, y_trspl, y_tespl = TrTeSplit(X_train, y_train, 0.33)
```

In [14]:
```
#Implement Tree model with the choosen parameter

#numbers_sizes = (i*10**exp for exp in range(2, 9) for i in range(1, 10))
#for i in numbers_sizes:
model = DecisionTreeRegressor(random_state = 1, max_depth = 20, min_samples_split = 6, min_samples_leaf = 2)
model.fit(X_trspl, y_trspl)
y_pred = model.predict(X_tespl)

rmse = RMSE(y_tespl, y_pred)
print(rmse)

feat_importances = pd.Series(model.feature_importances_, index=X_trspl.columns)
feat_importances.nlargest(20).plot(kind='barh')
plt.title('Feature Important based on Decision Tree regressor')
plt.xlabel('feature scores')
plt.ylabel('feature names')
plt.show()
```

```
51318.32419808552
```