Regression

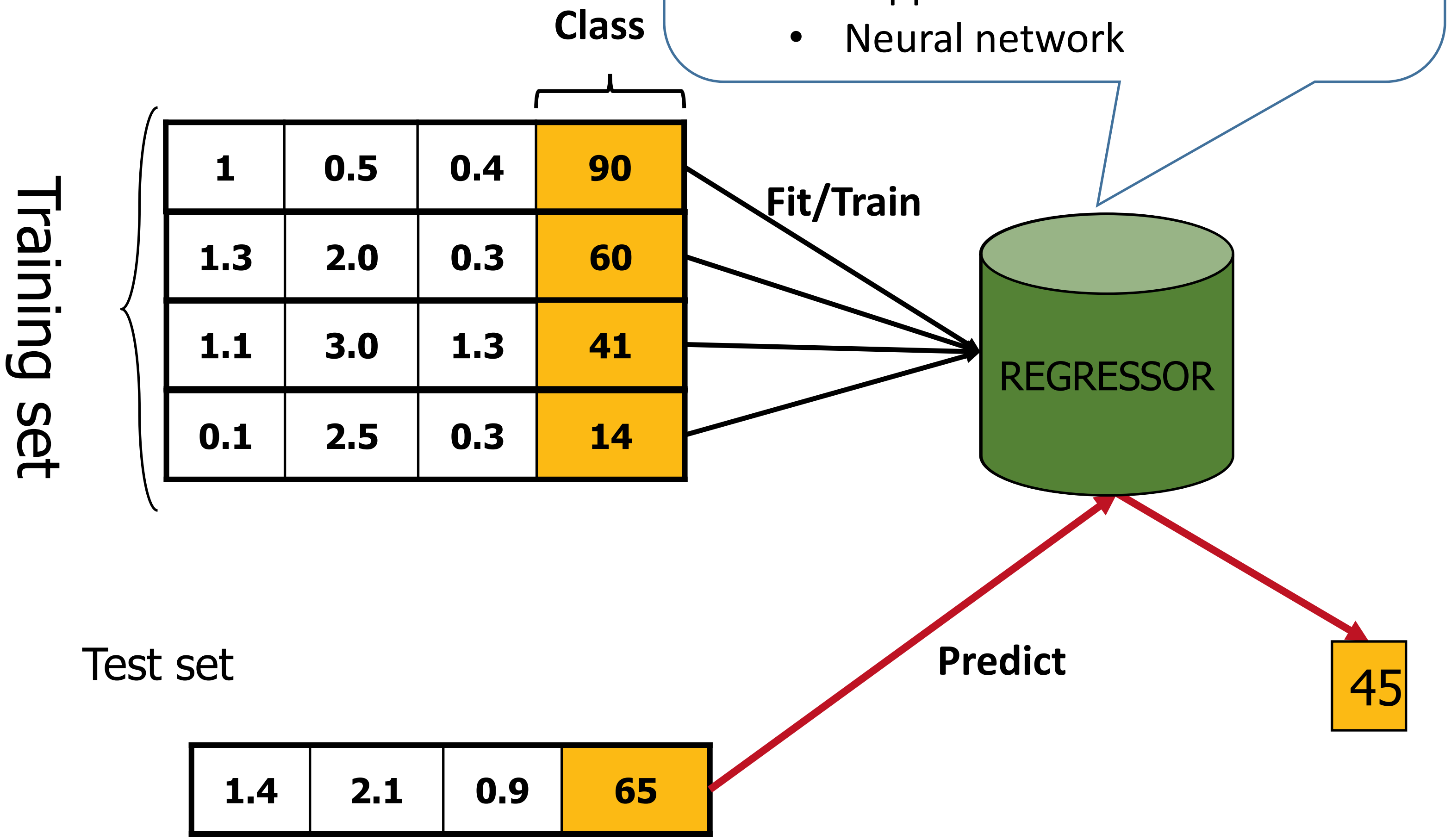
module 12

Regression

- Goal: Predict **numeric** target (outcome) variable
  - Examples: age, price, etc…
- Each row is a case (customer, student, applicant)
- Each column is a variable

Regression

- Some regressors are interpretable
  - OLS regression, Lasso
  - Decision Tree Regressors
- Some aren't
  - Support vector machines
  - Neural network



Training set

			Class
1	0.5	0.4	90
1.3	2.0	0.3	60
1.1	3.0	1.3	41
0.1	2.5	0.3	14

Test set

1.4	2.1	0.9	65

Regression for Data Exploration

Case study similar to project

Today’s data set

- https://www.kaggle.com/uciml/adult-census-income
- This data was extracted from the 1994 Census bureau database by Ronny Kohavi and Barry Becker (Data Mining and Visualization, Silicon Graphics). *The prediction task is to determine whether a person makes over $50K a year*.

- The cleaning is the same as in Module 9
- Today’s goal: **predict age**

The LASSO regressor

Linear prediction: $\widehat{y}_i = \mathbf{w} \cdot \mathbf{x}_i$

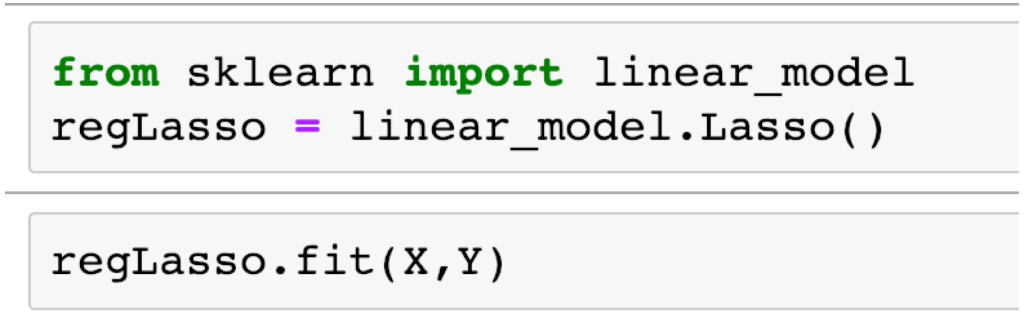
Minimize this function:

$$L = \underbrace{\sum_i \frac{1}{2}(\hat{y}_i - y_i)^2}_{\text{Error magnitude}} \quad + \quad \underbrace{\frac{1}{2}\lambda \overset{\sum_i |w_i|}{\boxed{|\mathbf{w}|_1}}}_{\text{Penalty for non-zero coefficients}}$$

Unlike regular regression, the LASSO regressor will try to minimize the number of predictors used.

LASSO in scikit learn

TRAIN

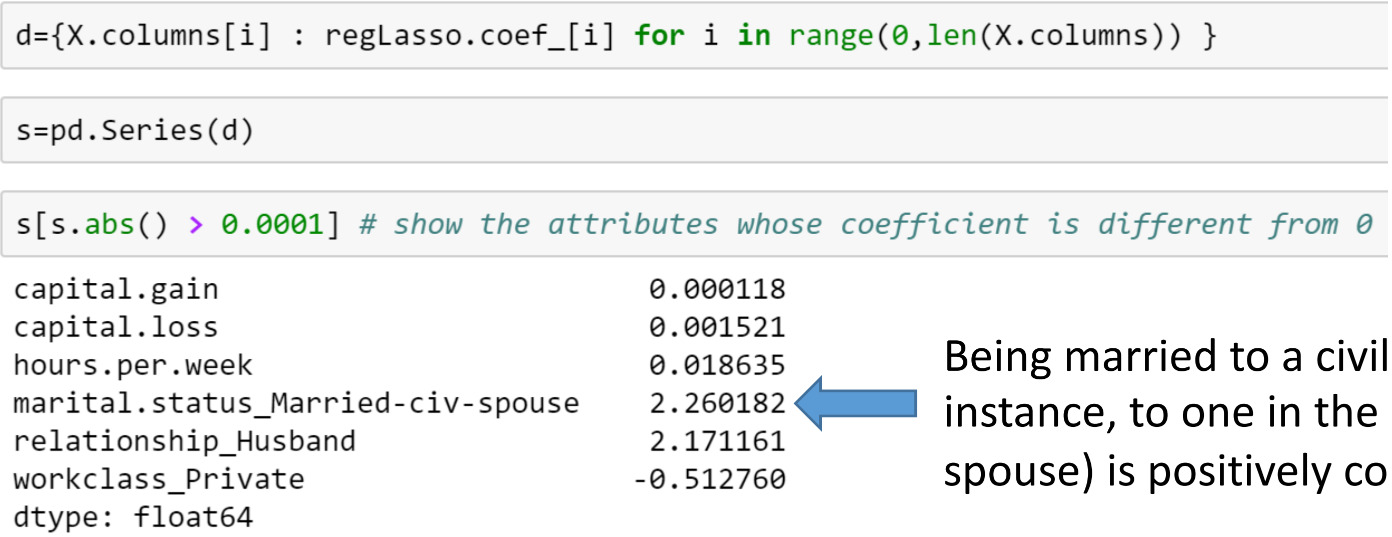


```
from sklearn import linear_model
regLasso = linear_model.Lasso()
```

```
regLasso.fit(X,Y)
```

INTERPRET

find the attributes with a non-zero coefficient



```
d={X.columns[i] : regLasso.coef_[i] for i in range(0,len(X.columns)) }
```

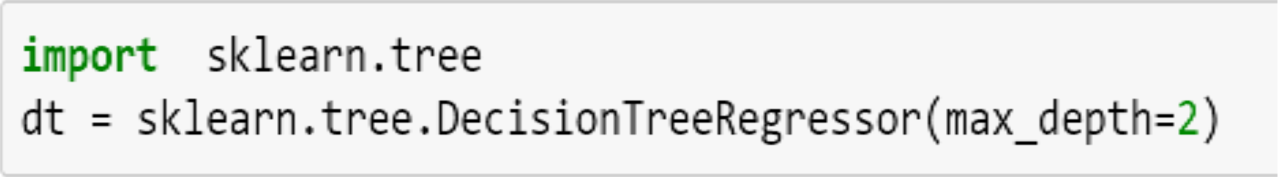
```
s=pd.Series(d)
```

```
s[s.abs() > 0.0001] # show the attributes whose coefficient is different from 0
```

```
capital.gain                         0.000118
capital.loss                         0.001521
hours.per.week                       0.018635
marital.status_Married-civ-spouse    2.260182
relationship_Husband                 2.171161
workclass_Private                   -0.512760
dtype: float64
```

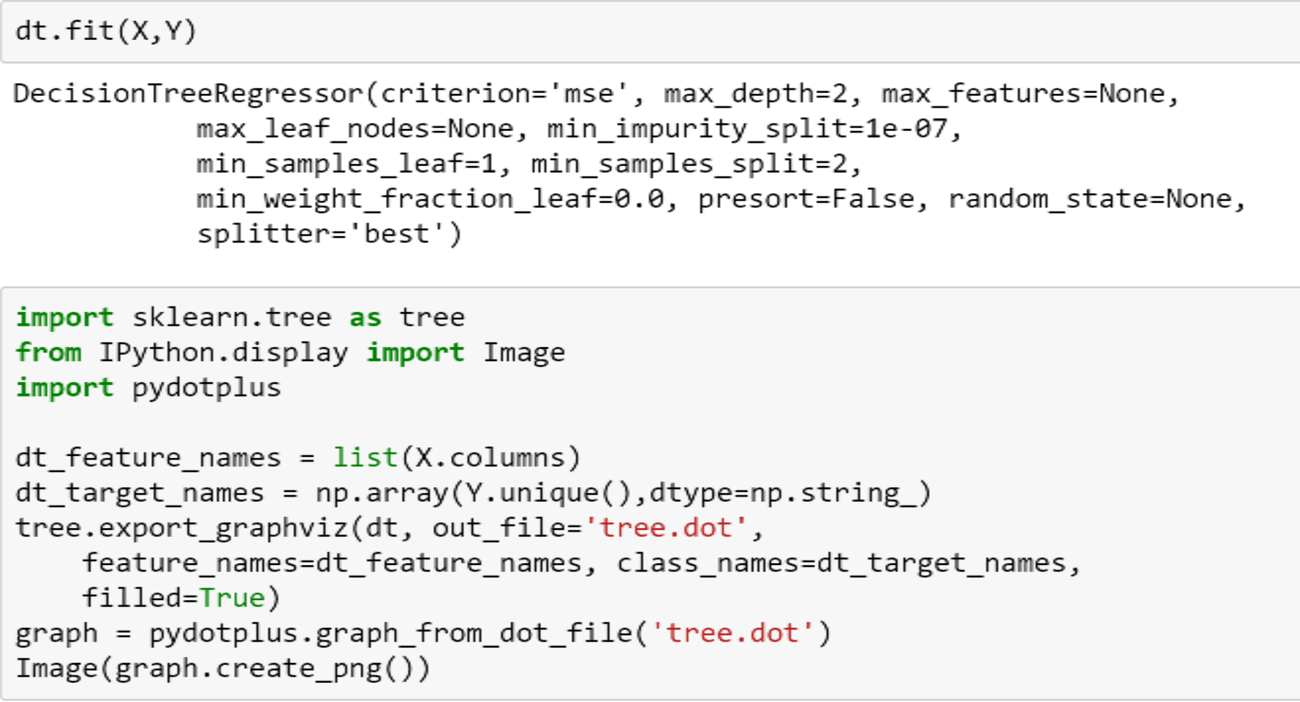
Being married to a civilian (rather than, for instance, to one in the armed forces, or AF-spouse) is positively correlated with age

Decision Tree Regressor



```
import  sklearn.tree
dt = sklearn.tree.DecisionTreeRegressor(max_depth=2)
```

Same as Decision Trees for Classification



```
dt.fit(X,Y)
```

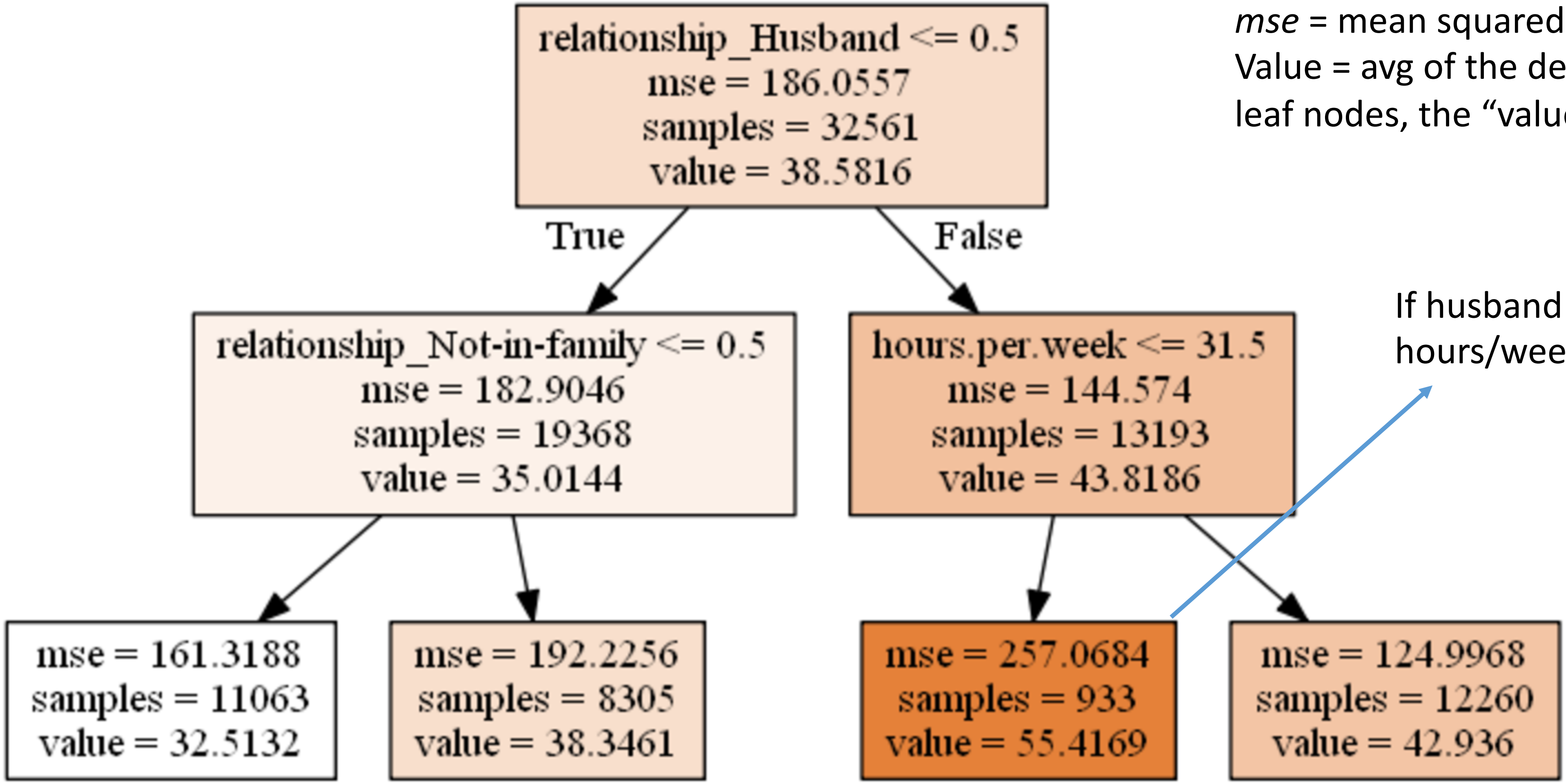
```
DecisionTreeRegressor(criterion='mse', max_depth=2, max_features=None,
           max_leaf_nodes=None, min_impurity_split=1e-07,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, presort=False, random_state=None,
           splitter='best')
```

```
import sklearn.tree as tree
from IPython.display import Image
import pydotplus

dt_feature_names = list(X.columns)
dt_target_names = np.array(Y.unique(),dtype=np.string_)
tree.export_graphviz(dt, out_file='tree.dot',
    feature_names=dt_feature_names, class_names=dt_target_names,
    filled=True)
graph = pydotplus.graph_from_dot_file('tree.dot')
Image(graph.create_png())
```

Decision Tree Regressor

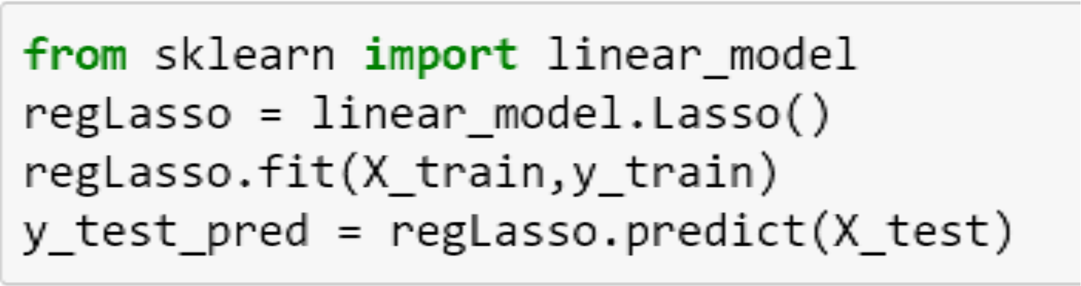
mse = mean squared error
Value = avg of the dependent variable (in leaf nodes, the "value" is the prediction)



If husband working <= 31 hours/week, predict 55.4 years old

Regression for Prediction

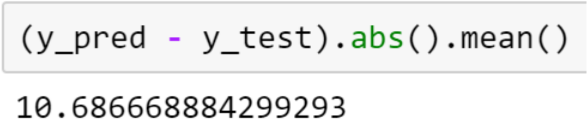
Regression for prediction



```
from sklearn import linear_model
regLasso = linear_model.Lasso()
regLasso.fit(X_train,y_train)
y_test_pred = regLasso.predict(X_test)
```

How to measure predictive performance?

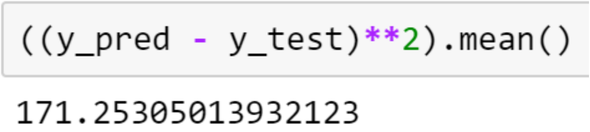
- Mean absolute deviation (MAD) = $\frac{1}{n}\sum_{i=0}^{n}\left|y_{actual}(i) - y_{pred}(i)\right|$



```
(y_pred - y_test).abs().mean()
```

```
10.686668884299293
```

- Mean squared error (MSE) = $\frac{1}{n}\sum_{i=0}^{n}\left(y_{actual}(i) - y_{pred}(i)\right)^2$



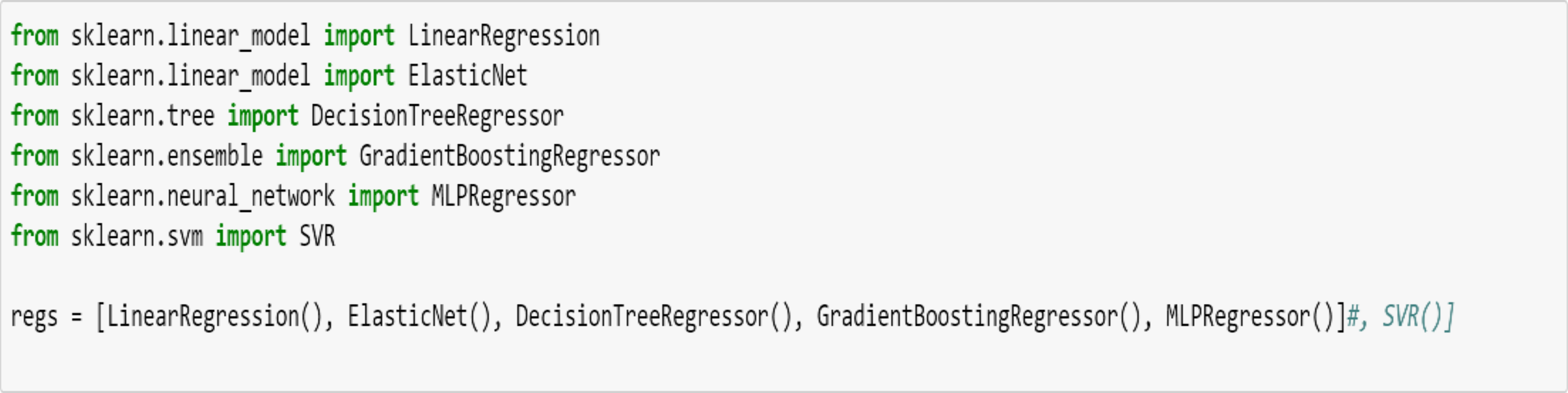
```
((y_pred - y_test)**2).mean()
```

```
171.25305013932123
```

- …

Finding the best regressor

Here are a few regressors

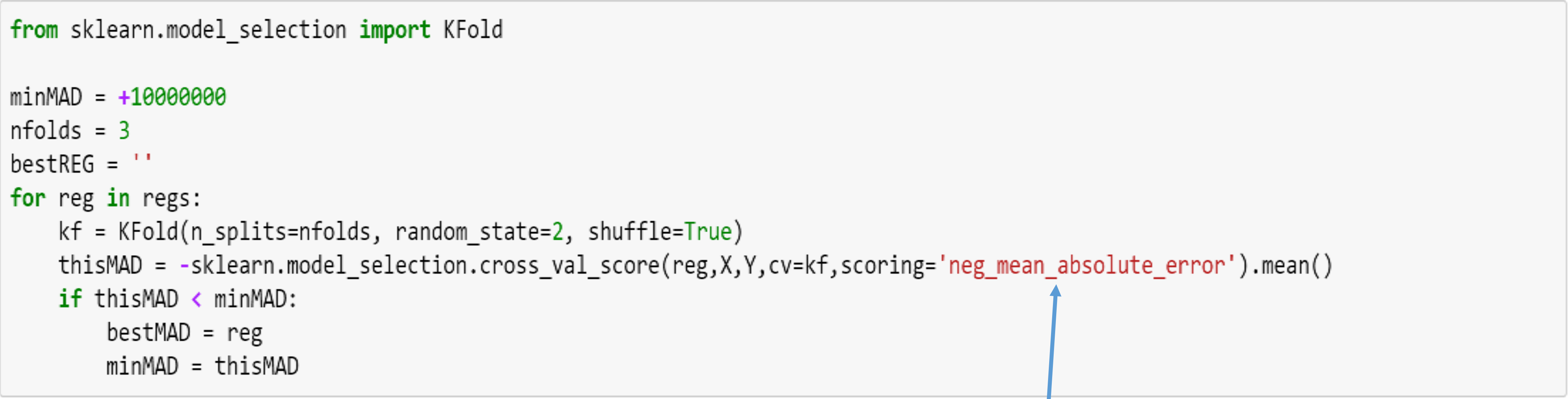


```python
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import ElasticNet
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.svm import SVR

regs = [LinearRegression(), ElasticNet(), DecisionTreeRegressor(), GradientBoostingRegressor(), MLPRegressor()]#, SVR()]
```

Finding the best regressor

We can use Cross-validation to find the best one



```
from sklearn.model_selection import KFold

minMAD = +10000000
nfolds = 3
bestREG = ''
for reg in regs:
    kf = KFold(n_splits=nfolds, random_state=2, shuffle=True)
    thisMAD = -sklearn.model_selection.cross_val_score(reg,X,Y,cv=kf,scoring='neg_mean_absolute_error').mean()
    if thisMAD < minMAD:
        bestMAD = reg
        minMAD = thisMAD
```

Other scoring methods are reported here