

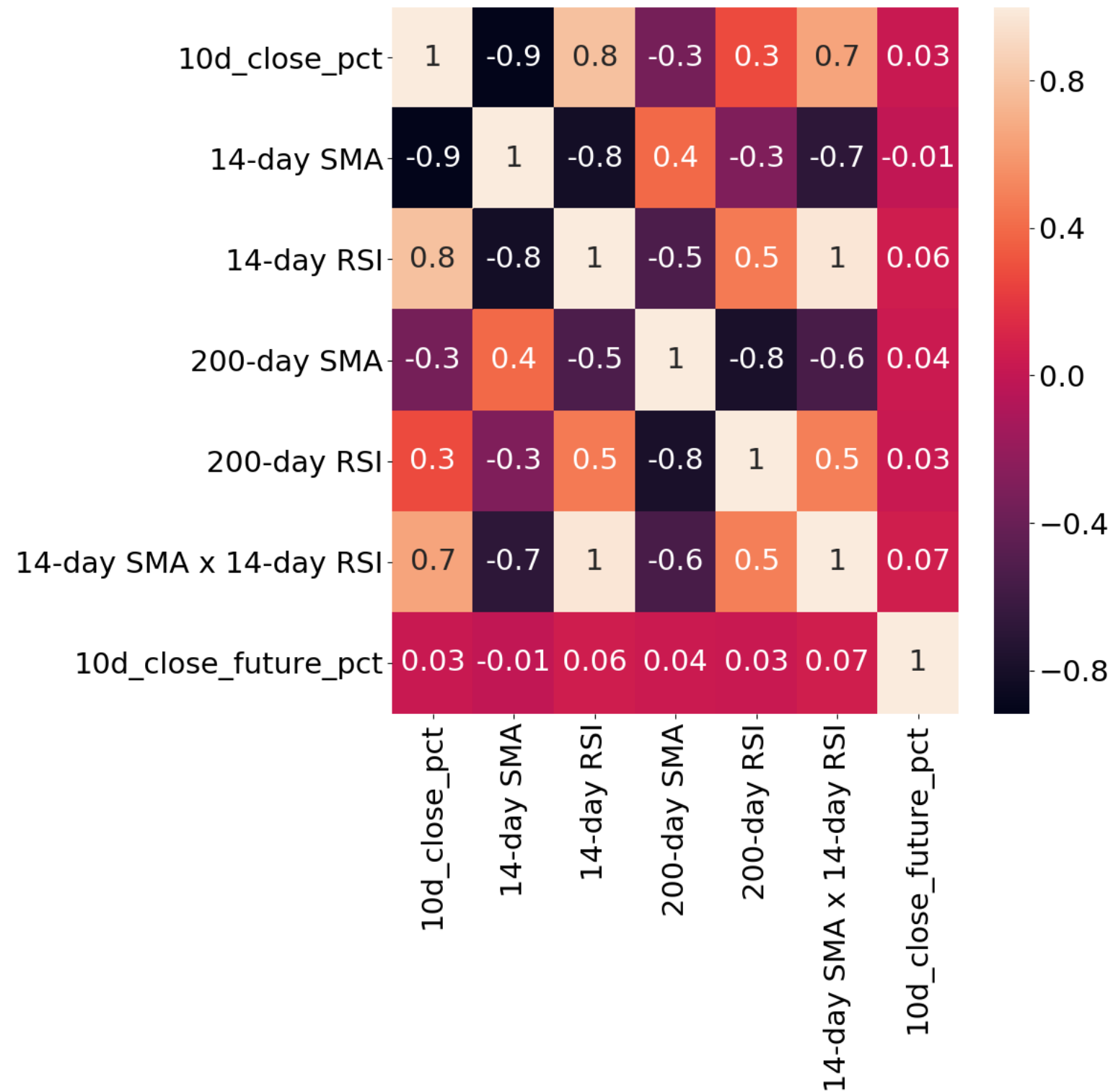


MACHINE LEARNING FOR FINANCE IN PYTHON

# Engineering features

Nathan George

Data Science Professor





# One problem with linear models

```
# add non-linear interaction term for a linear model  
SMAxRSI = amd_df['14-day SMA'] * amd_df['14-day RSI']
```

Some models that don't require manually creating interaction features:

Decision-tree-based models

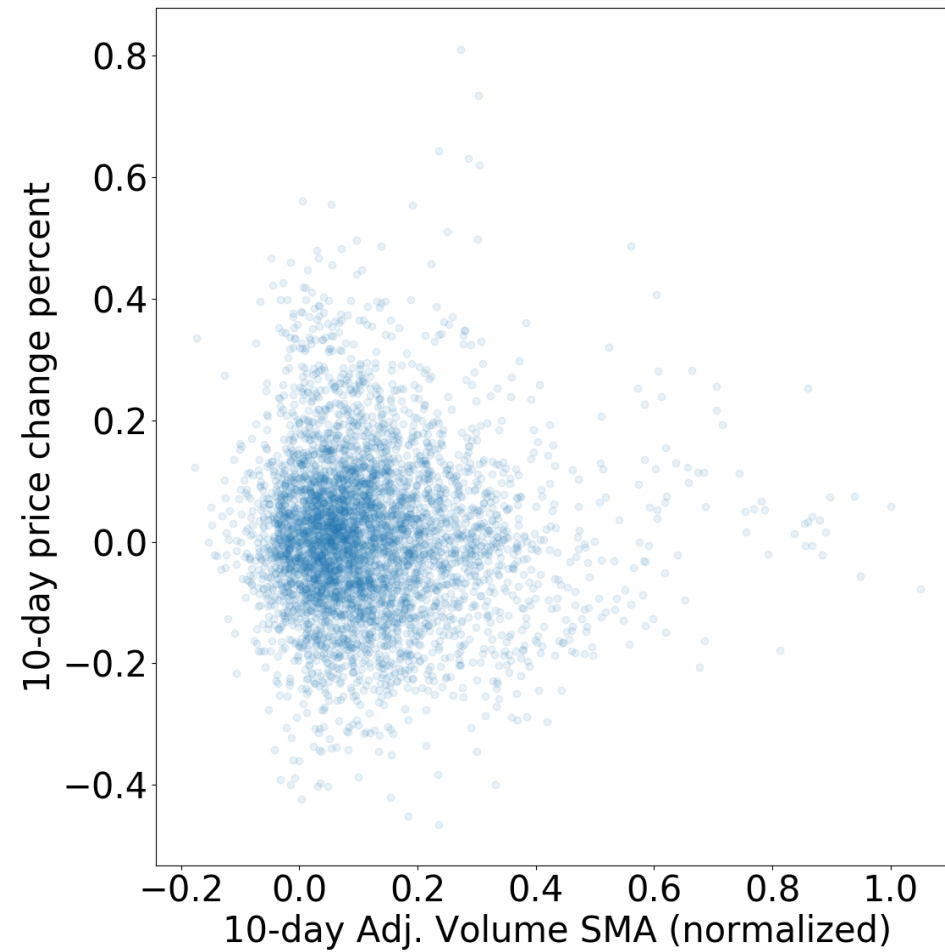
- Random forests
- Gradient boosting

Others

- neural networks

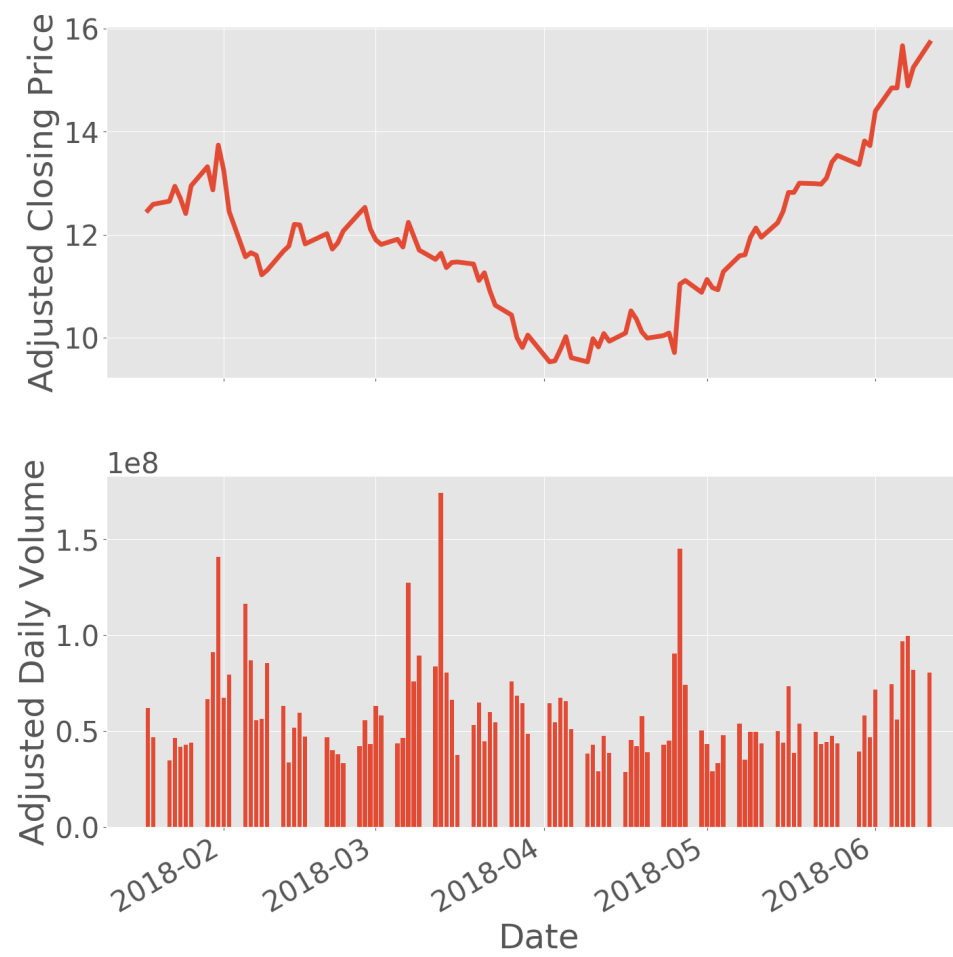


# Feature engineering





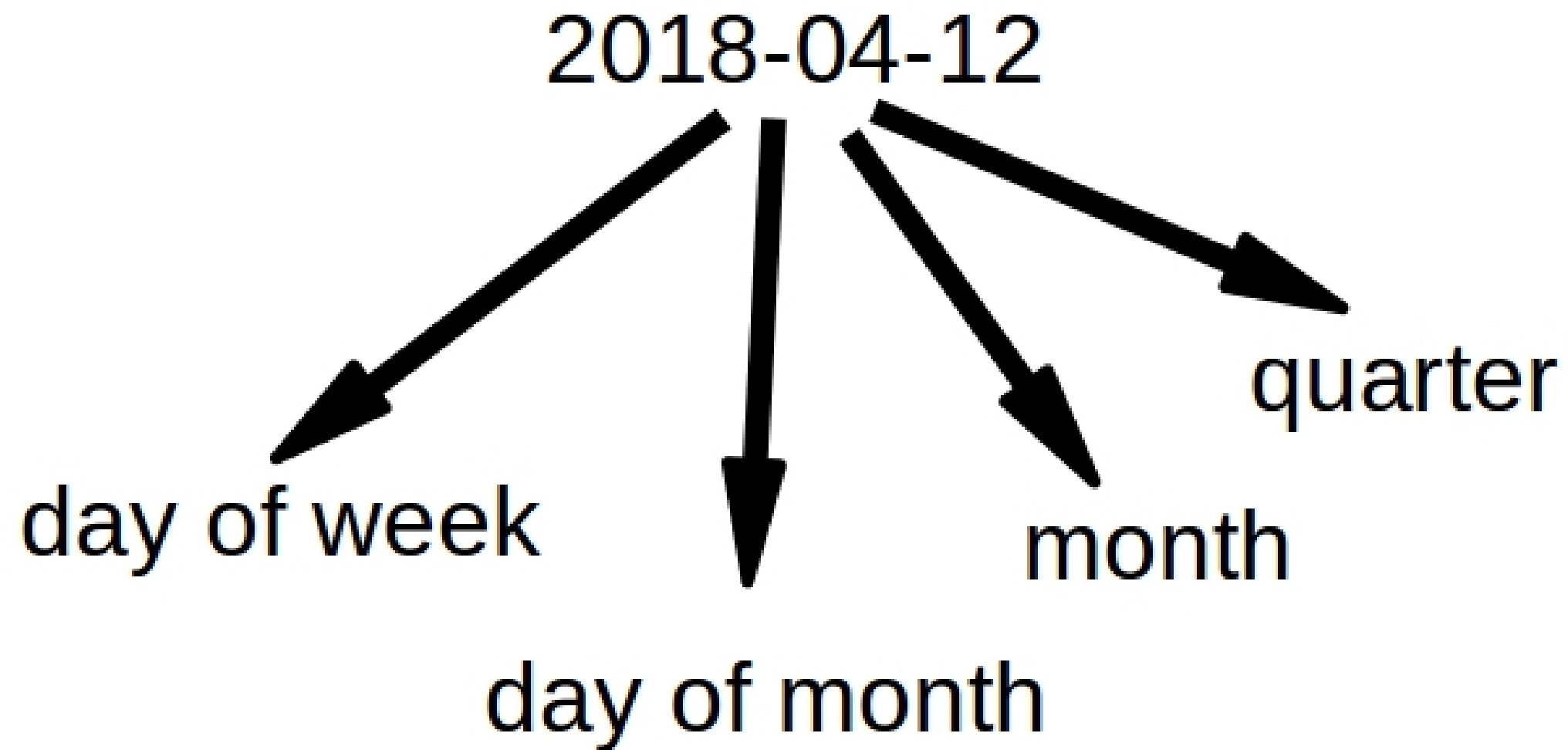
# Volume







# Datetime feature engineering





# Extracting the day of week

```
print(amd_df.index.dayofweek)
```

```
Int64Index([2, 3, 4, 0, 1, 2, 3, 4, 0, 1,  
           ...,  
           1, 2, 3, 4, 0, 1, 2, 3, 4, 0],  
           dtype='int64', name='Date', length=4807)
```



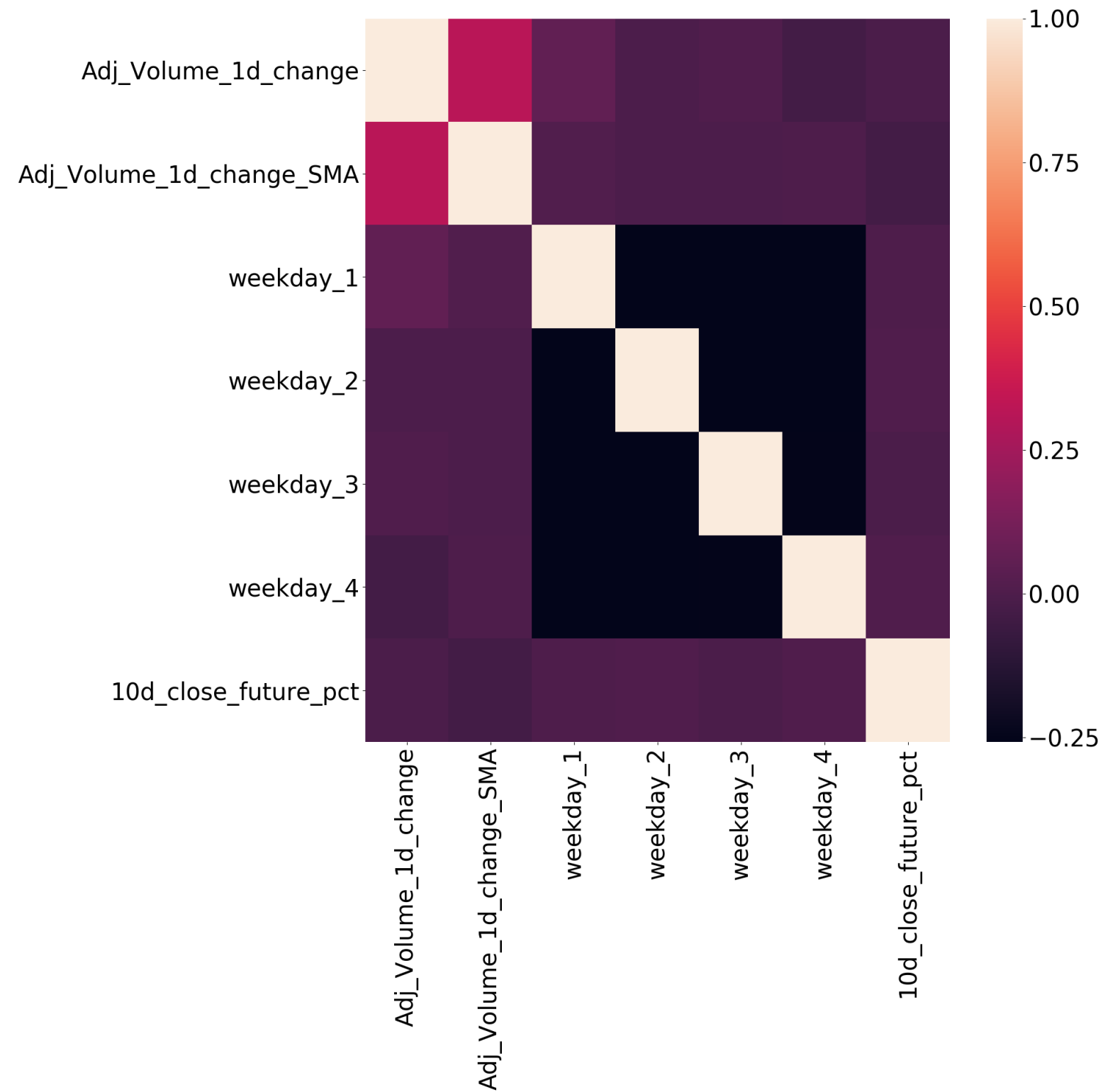


# Dummies

```
days_of_week = pd.get_dummies(amd_df.index.dayofweek,  
                               prefix='weekday',  
                               drop_first=True)
```

```
print(days_of_week.head())
```

	weekday_1	weekday_2	weekday_3	weekday_4
Date				
2018-04-10	1	0	0	0
2018-04-11	0	1	0	0
2018-04-12	0	0	1	0
2018-04-13	0	0	0	1
2018-04-16	0	0	0	0





## MACHINE LEARNING FOR FINANCE IN PYTHON

**Engineer some  
features!**



MACHINE LEARNING FOR FINANCE IN PYTHON

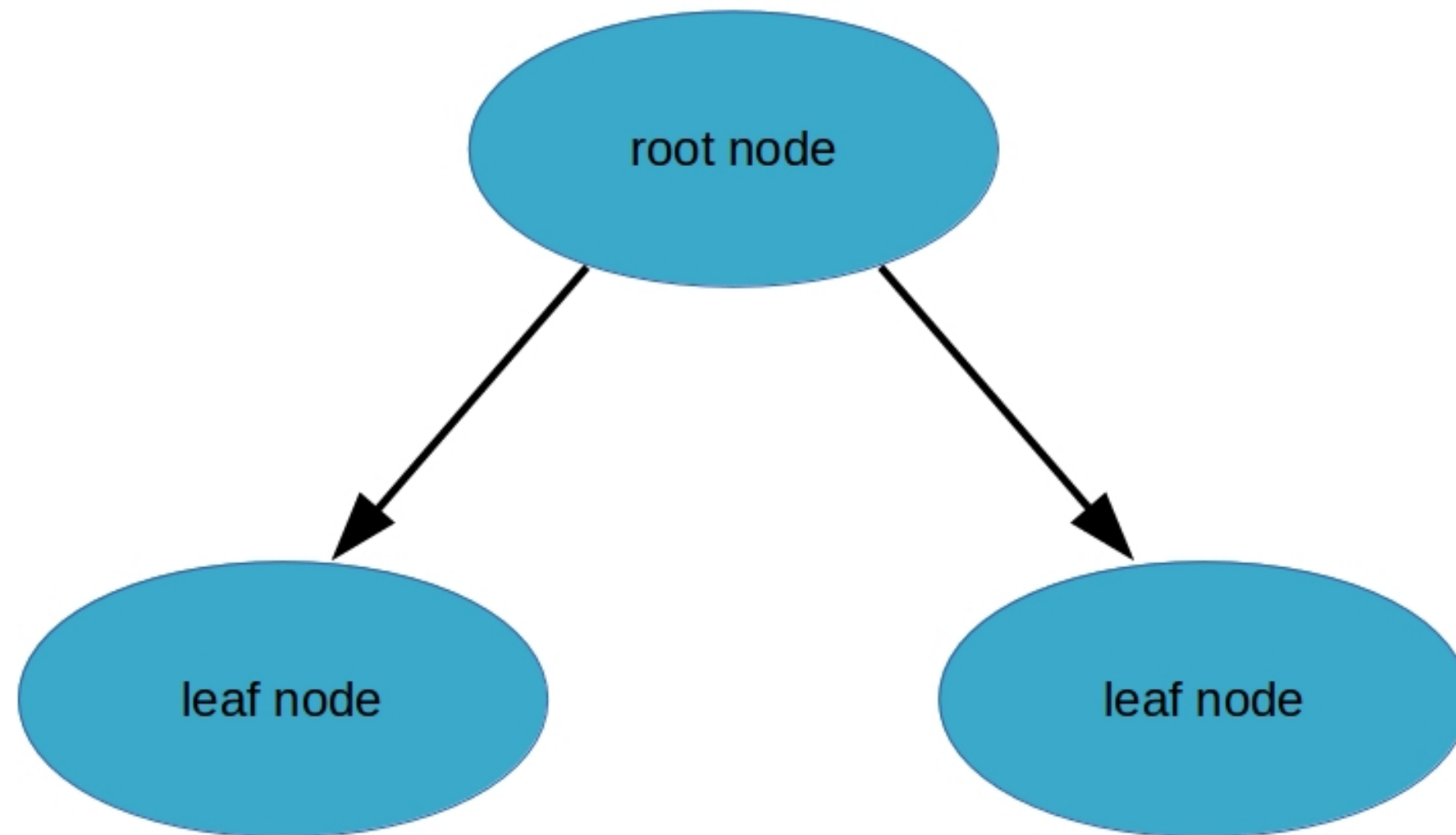
# Decision Trees

Nathan George

Data Science Professor

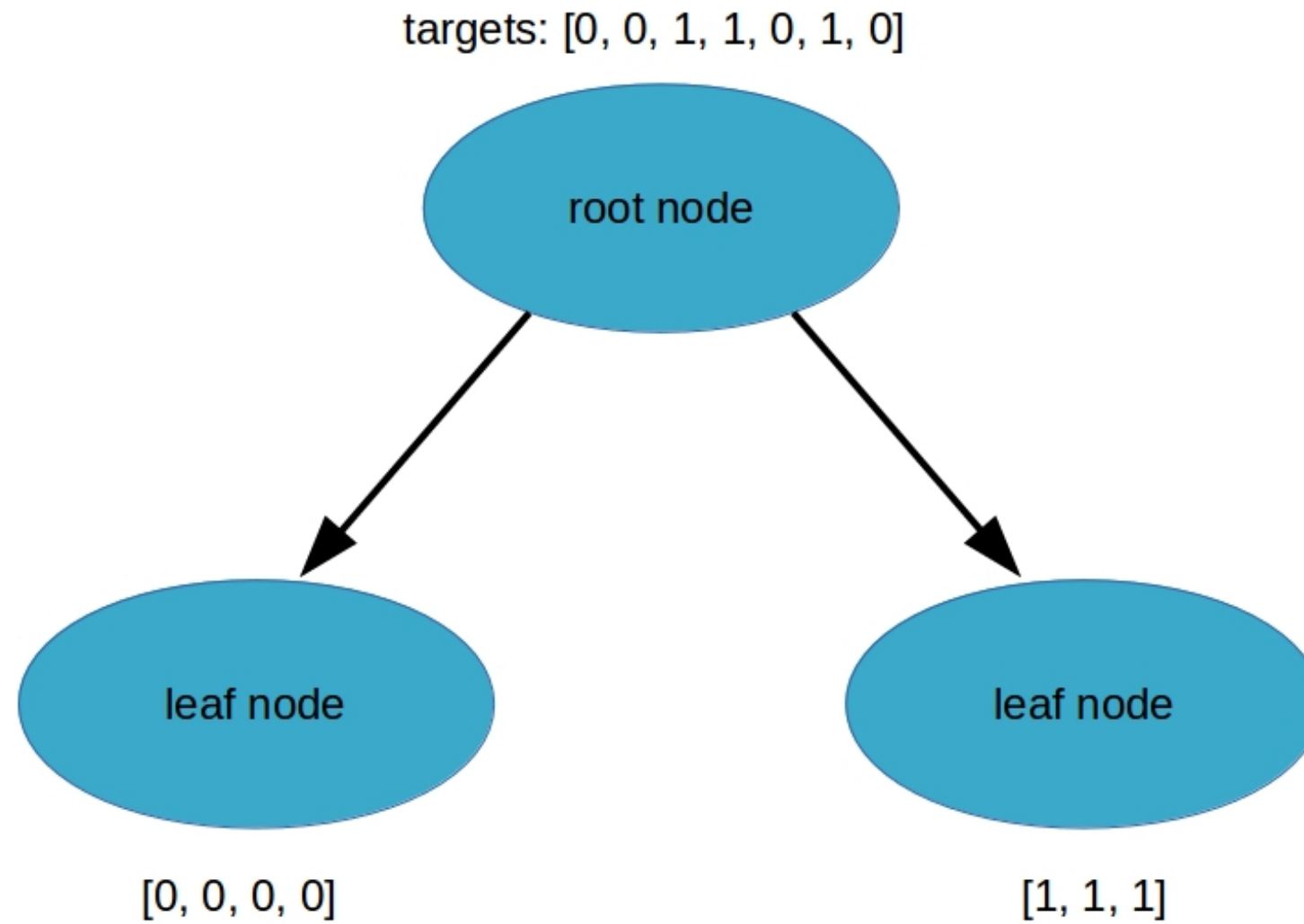


# Decision trees



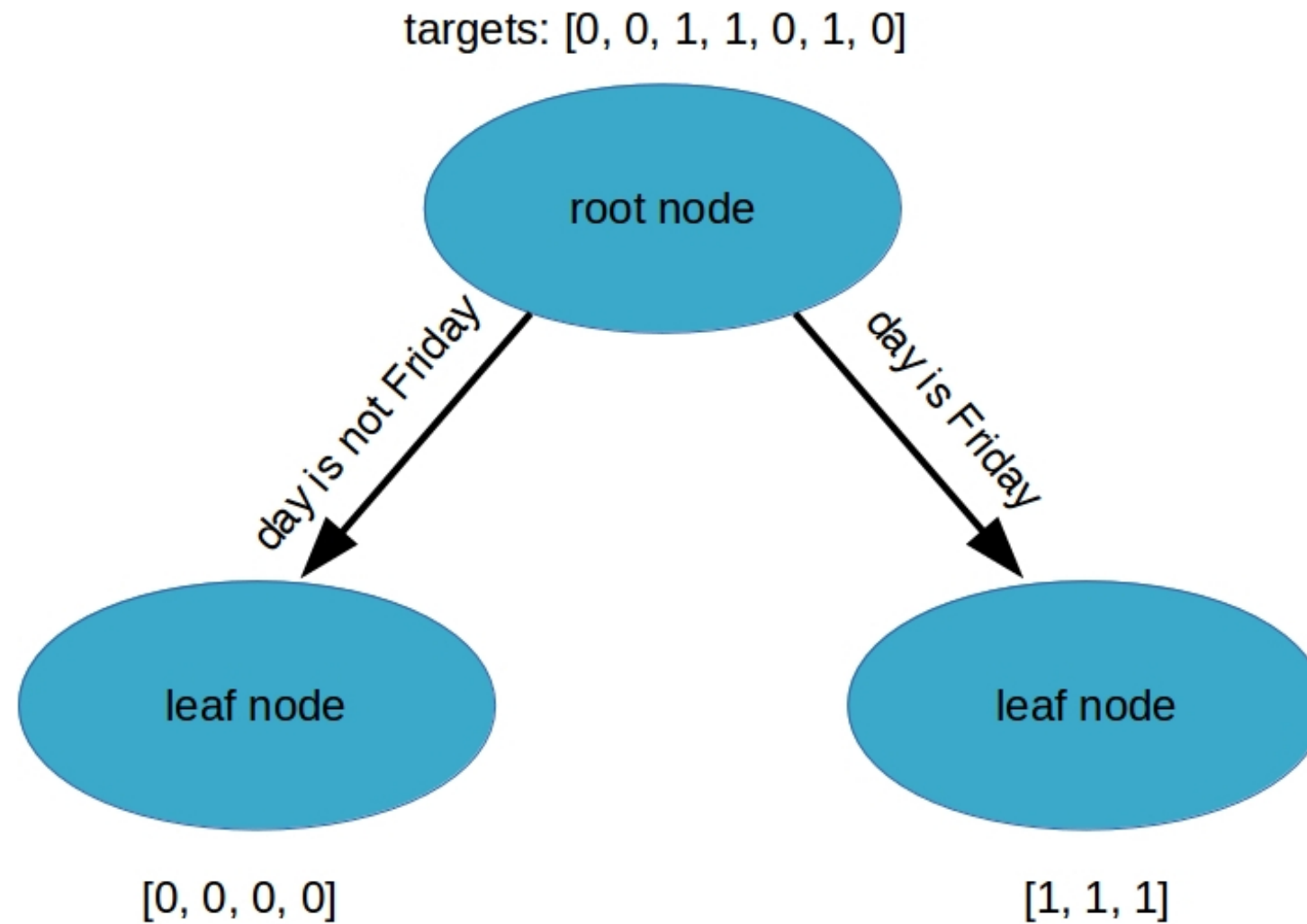


# Decision trees



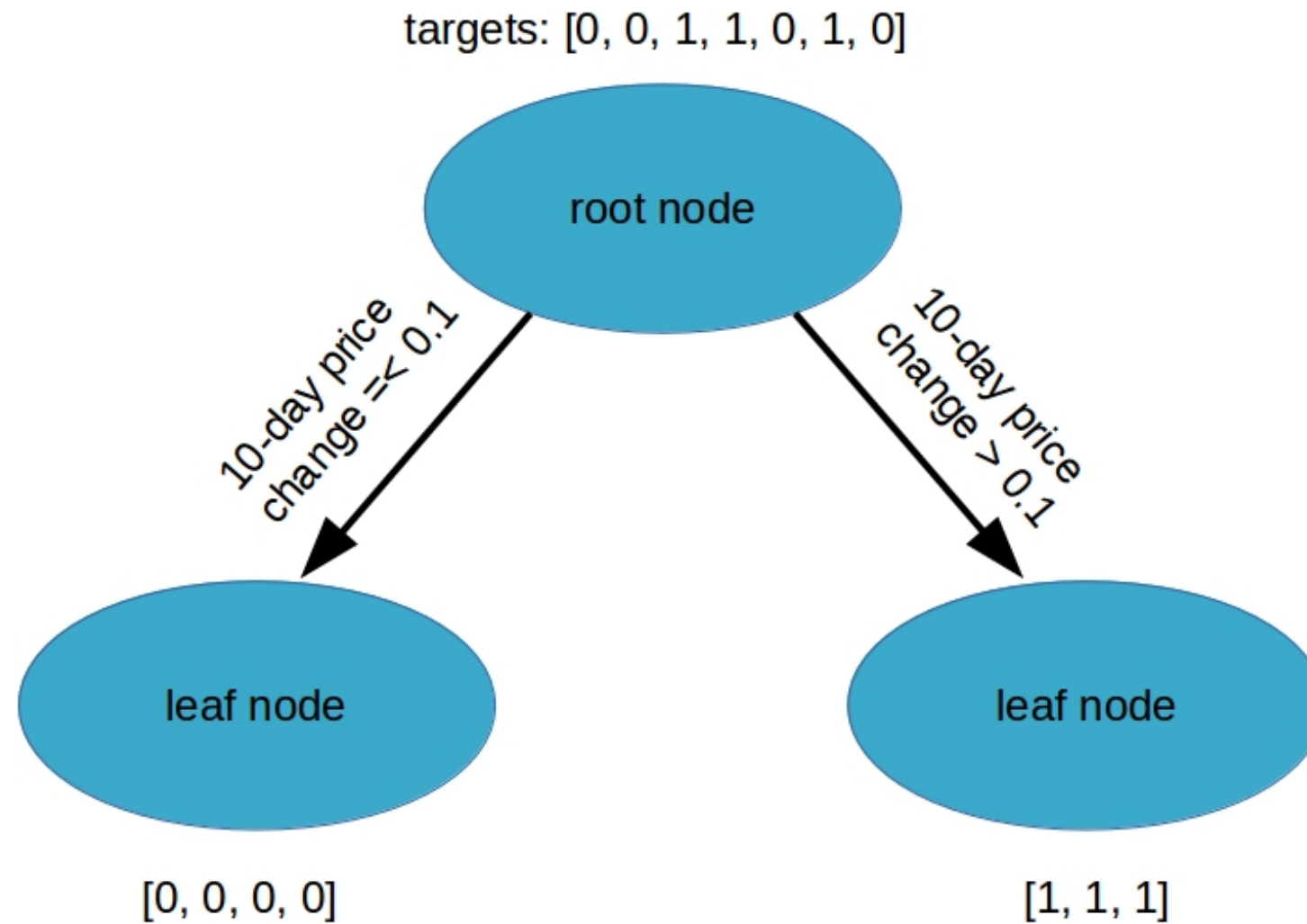


# Decision tree splits





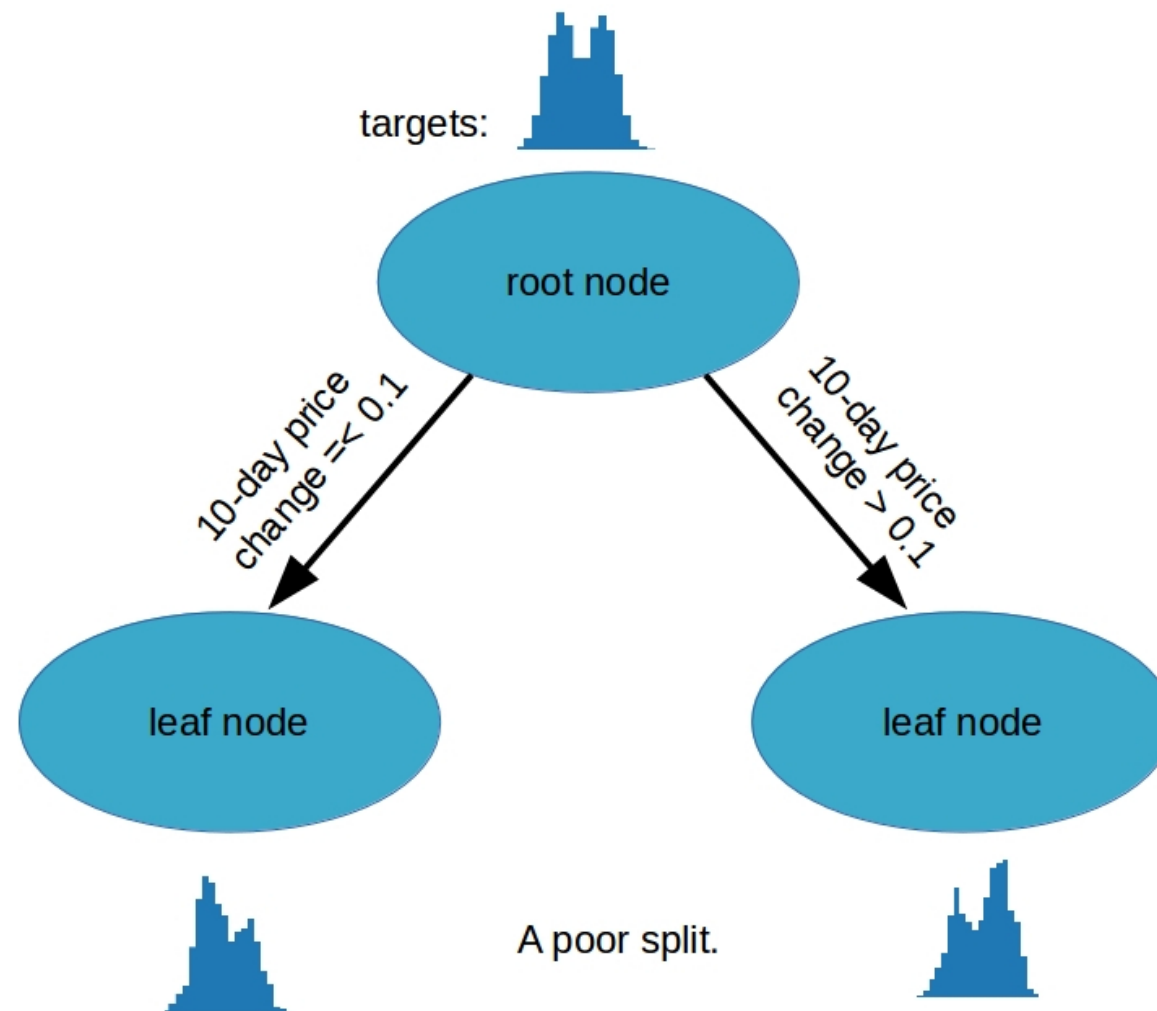
# Decision tree splits





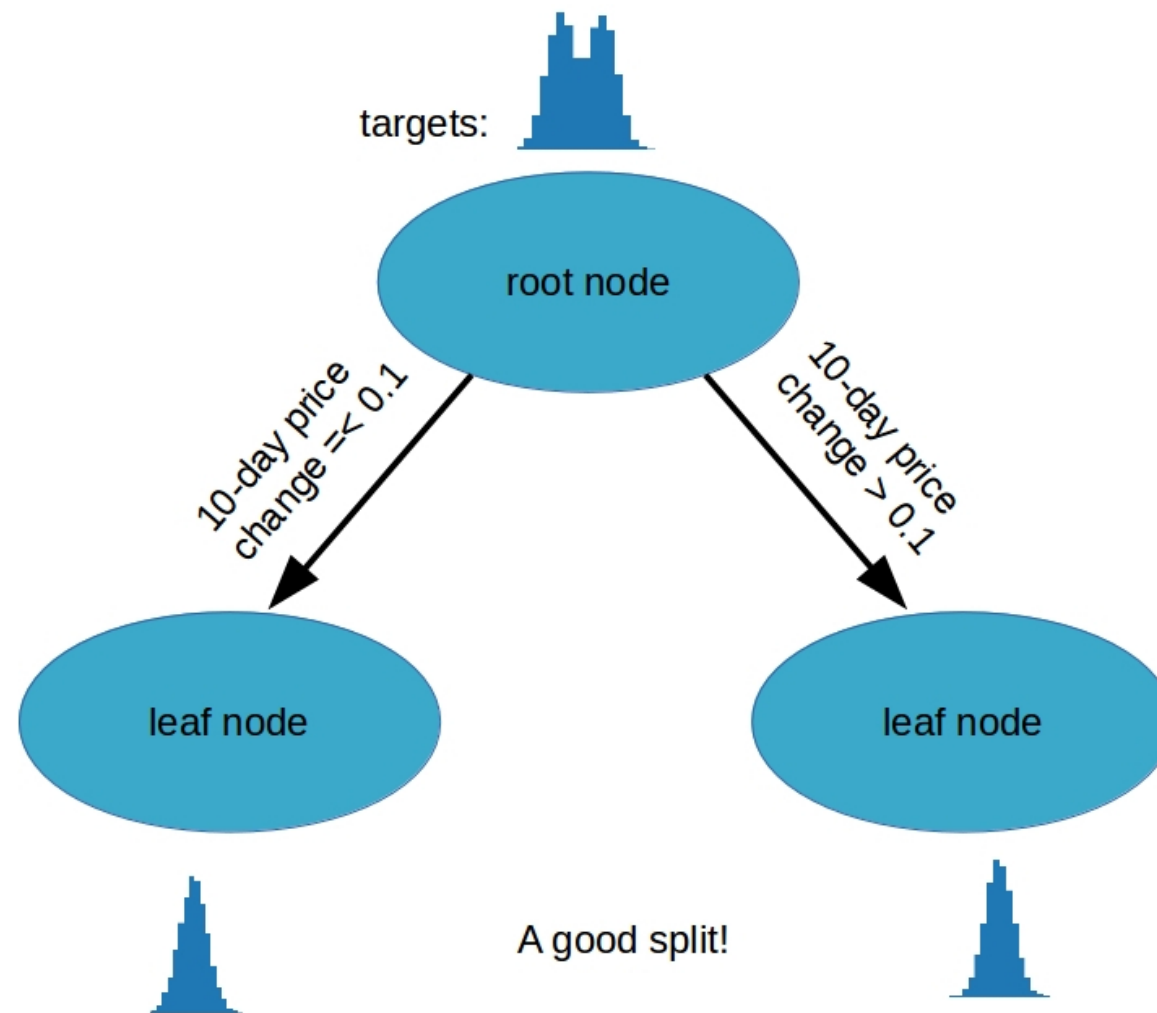


# Bad tree



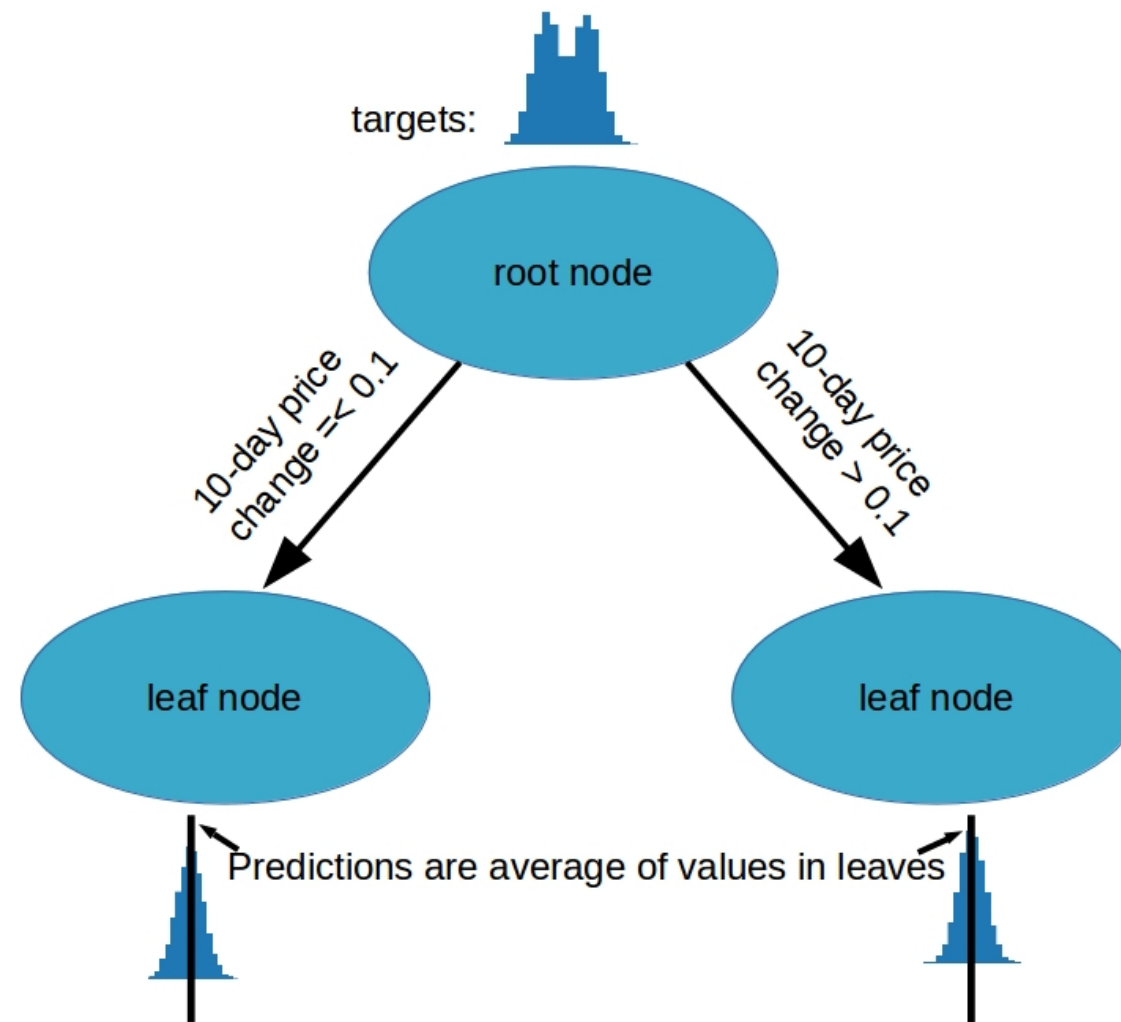


# Good tree





# Decision tree regression

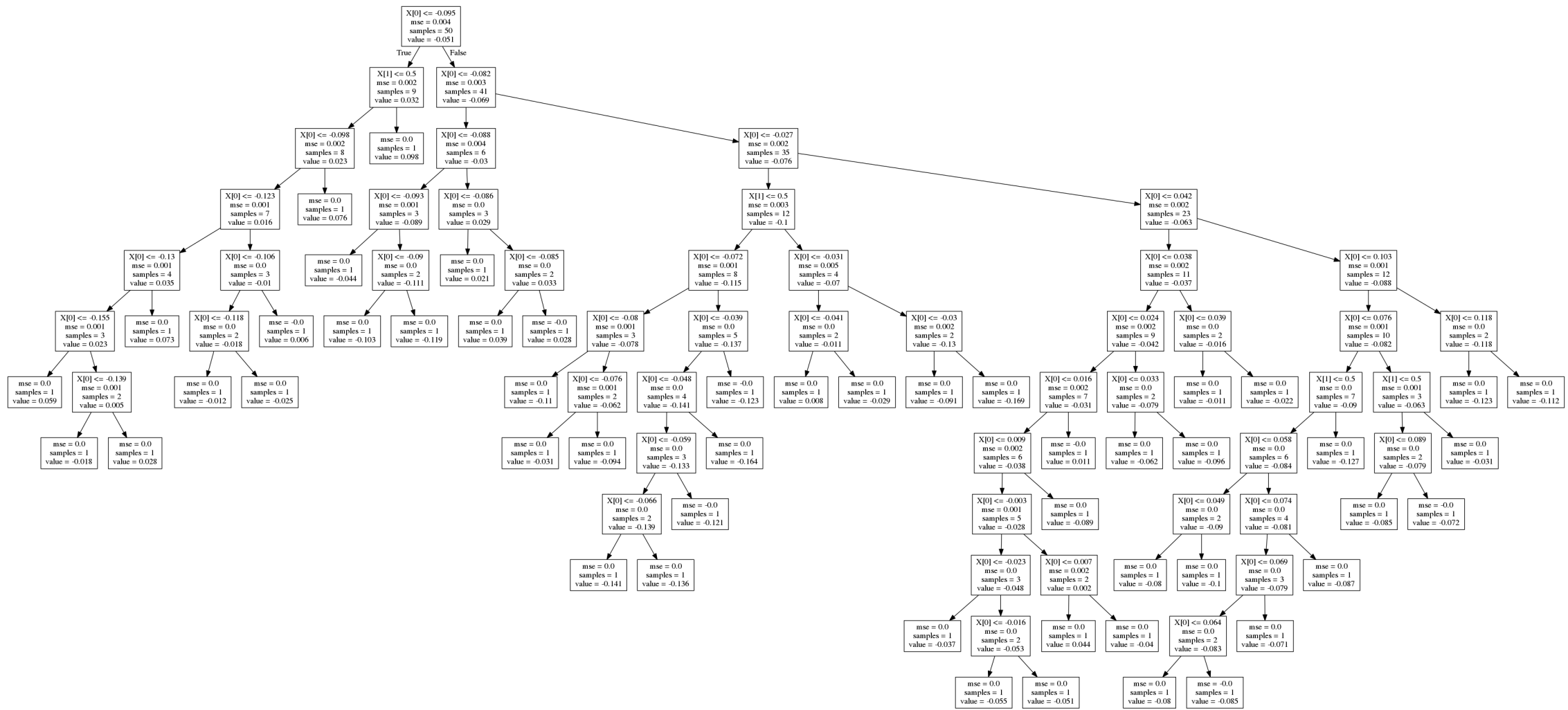




# Regression trees

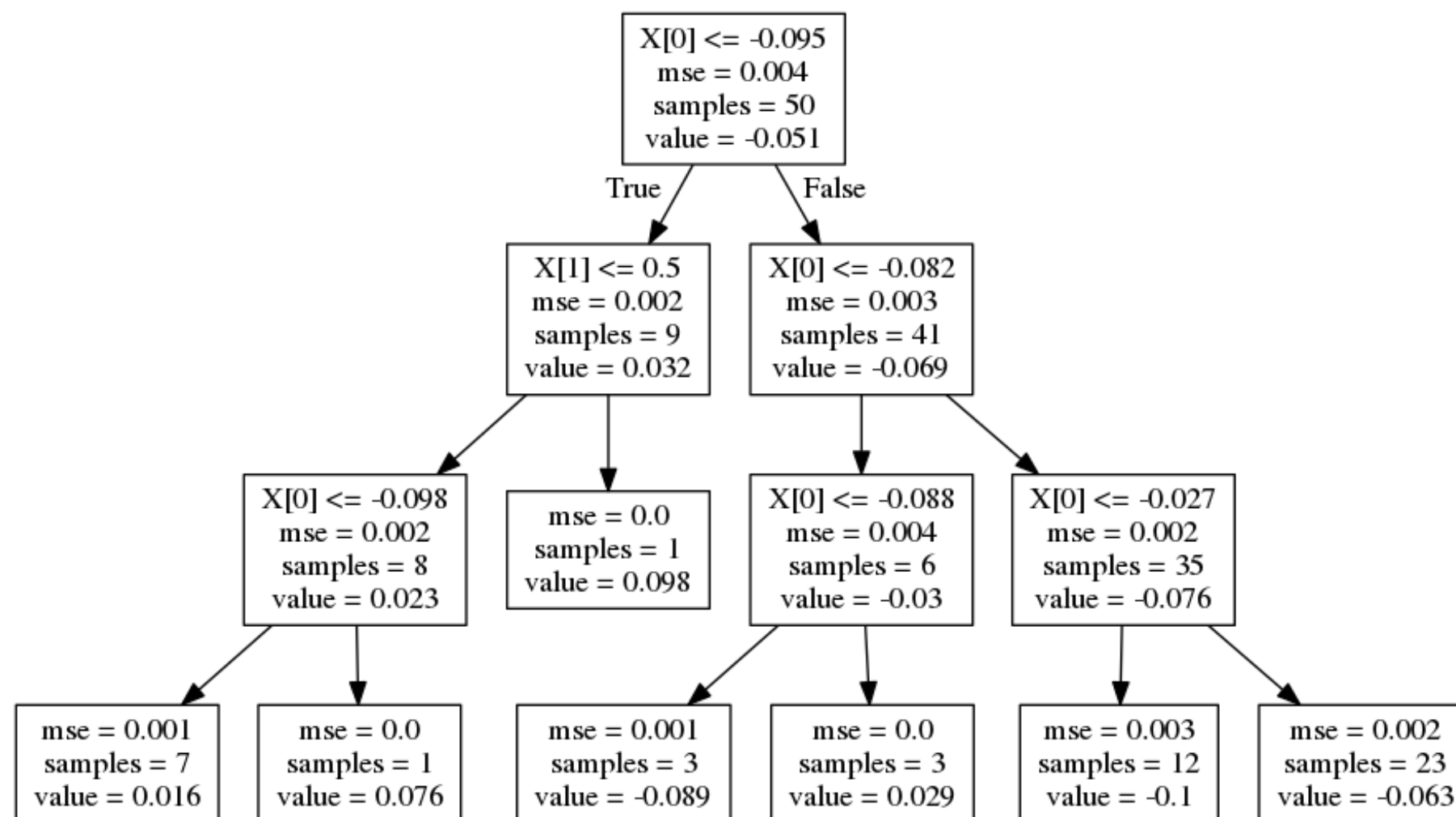
```
from sklearn.tree import DecisionTreeRegressor  
  
decision_tree = DecisionTreeRegressor(max_depth=5)  
  
decision_tree.fit(train_features, train_targets)
```

# Decision tree hyperparameters





# Max depth of 3



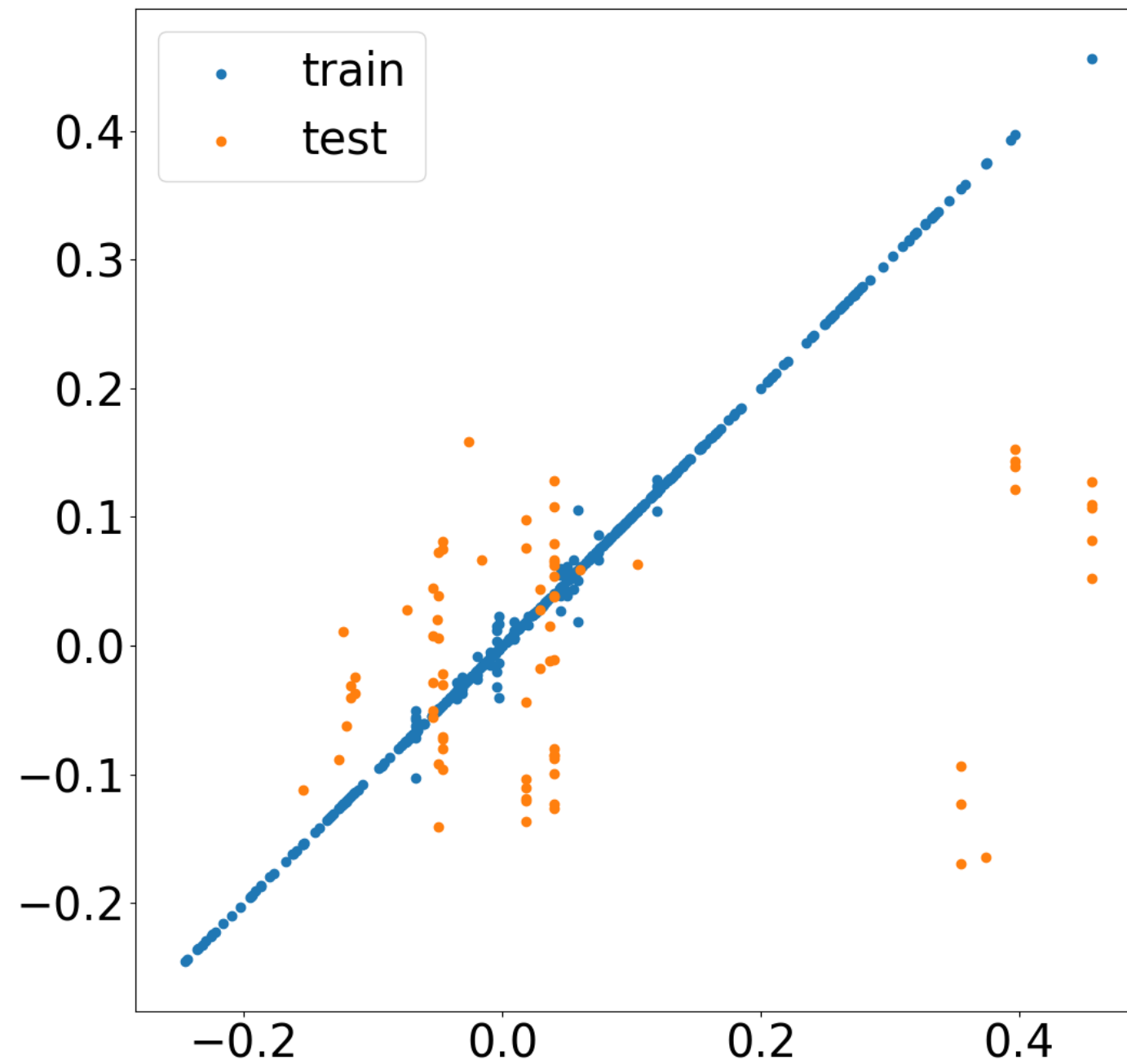


# Evaluate model

```
print(decision_tree.score(train_features, train_targets))  
print(decision_tree.score(test_features, test_targets))
```

```
0.6662215501032416  
-0.08917300191734268
```

```
train_predictions = decision_tree.predict(train_features)  
test_predictions = decision_tree.predict(test_features)  
plt.scatter(train_predictions, train_targets, label='train')  
plt.scatter(test_predictions, test_targets, label='test')  
plt.legend()  
plt.show()
```







## MACHINE LEARNING FOR FINANCE IN PYTHON

**Grow some trees!**

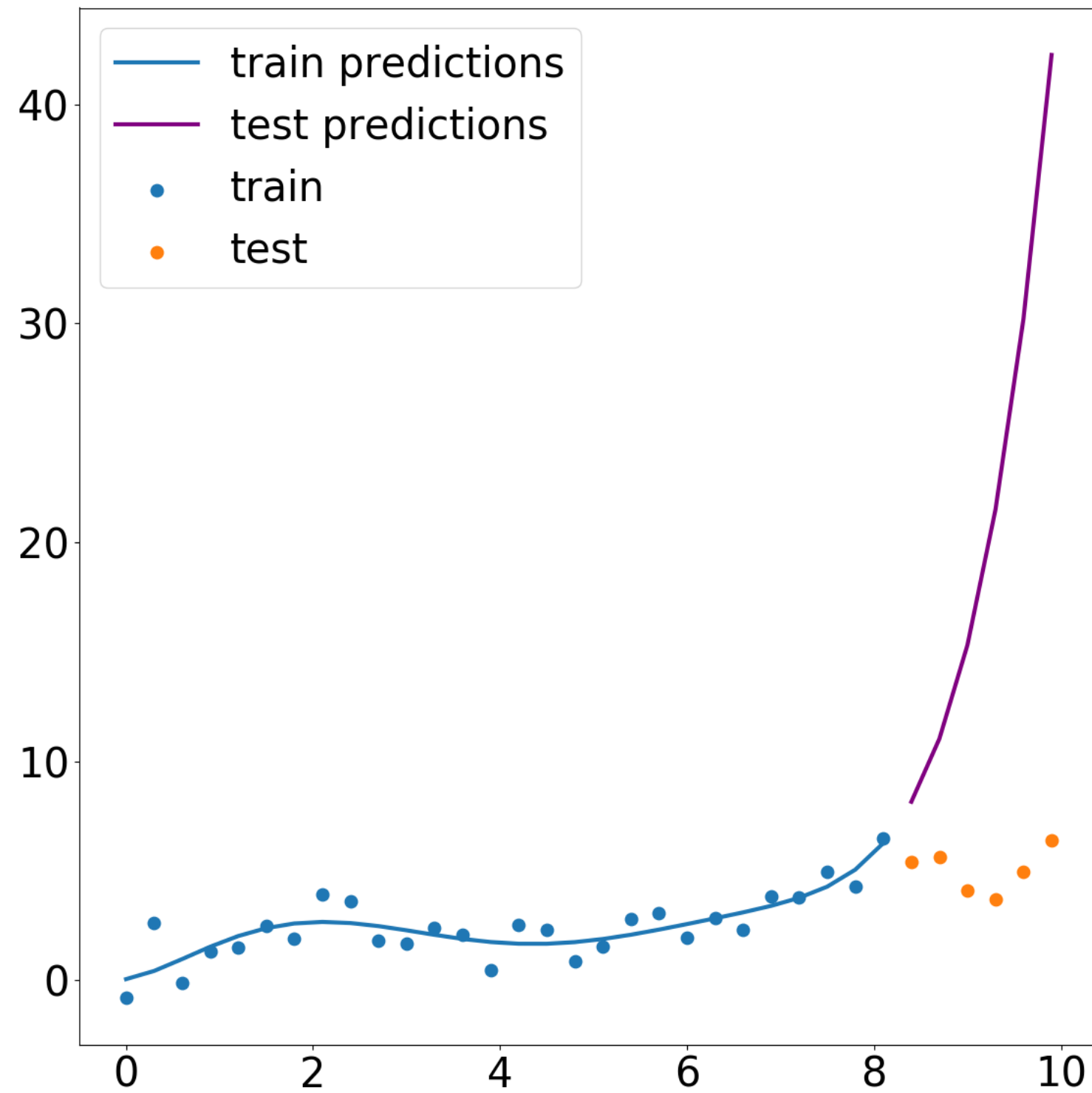


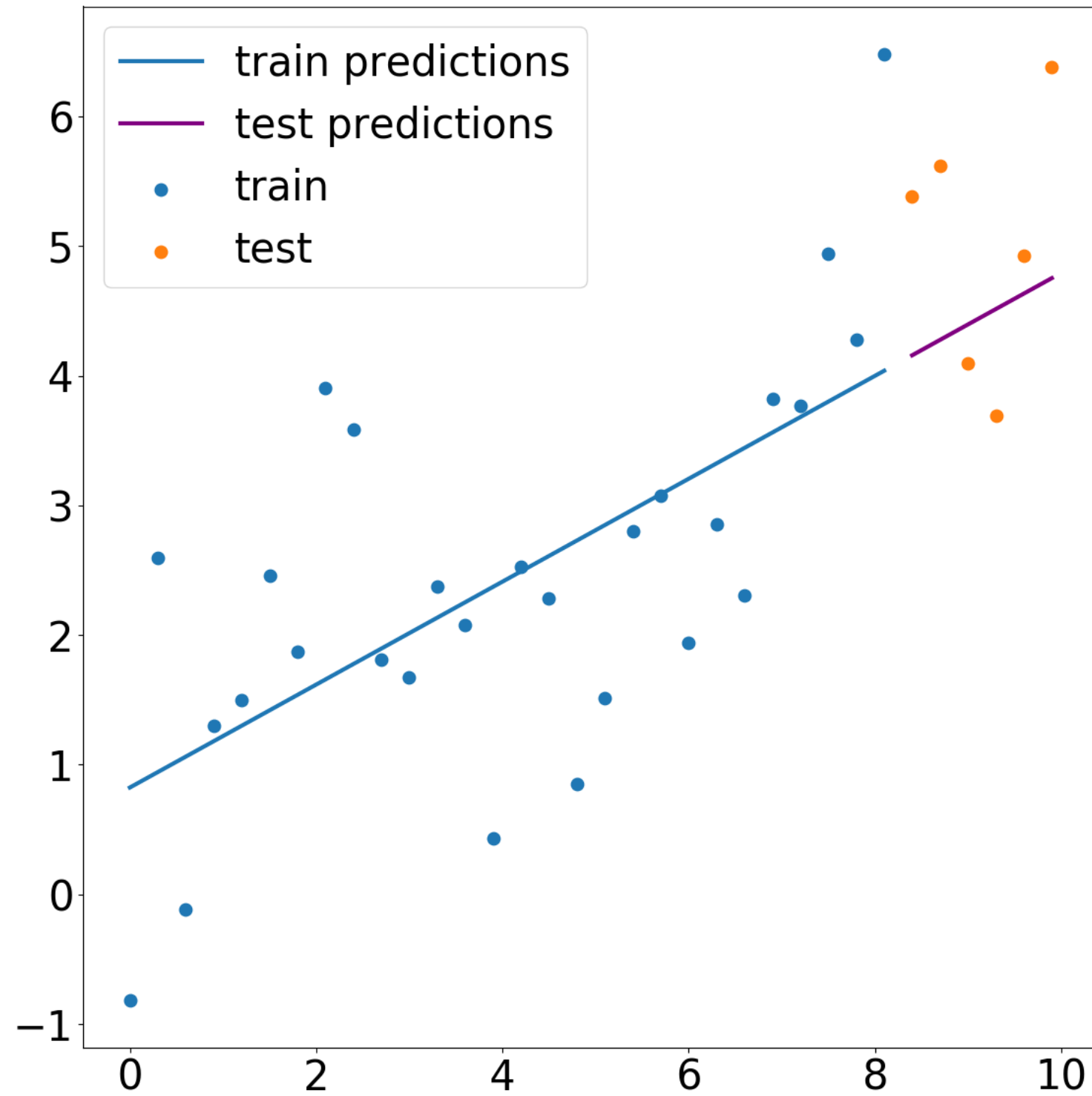
MACHINE LEARNING FOR FINANCE IN PYTHON

# Random forests

Nathan George

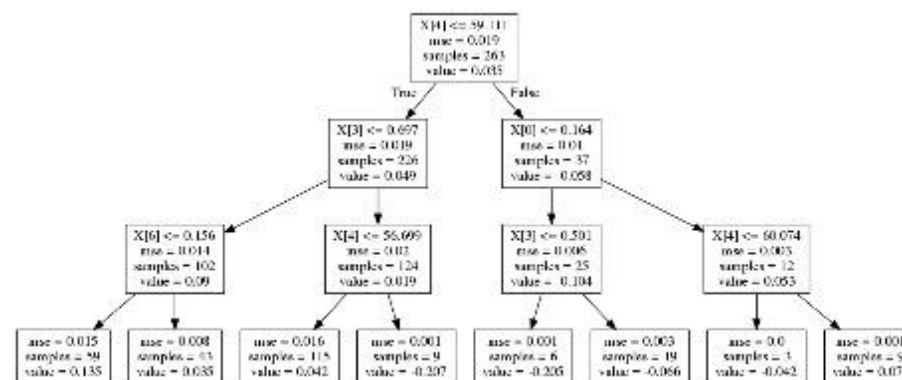
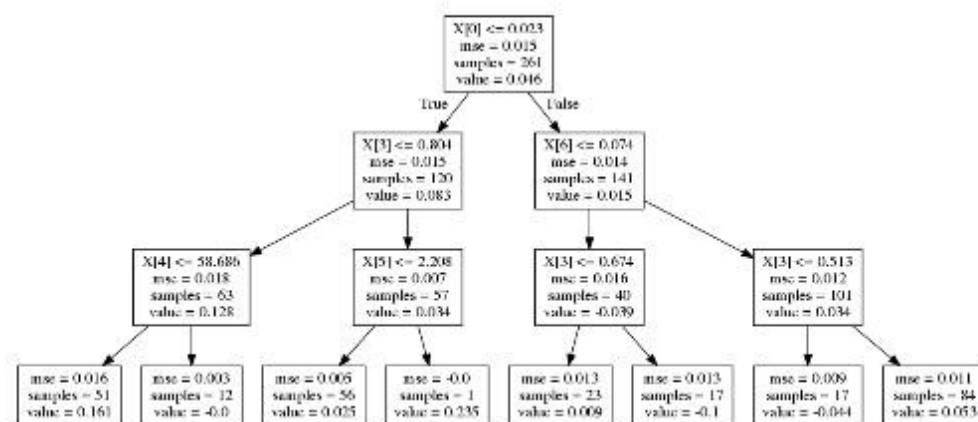
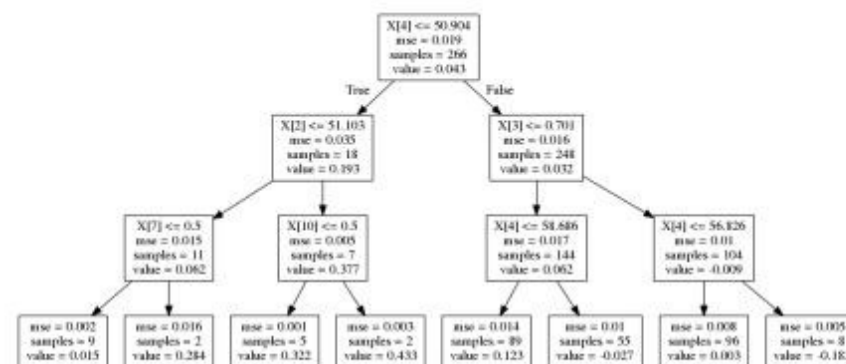
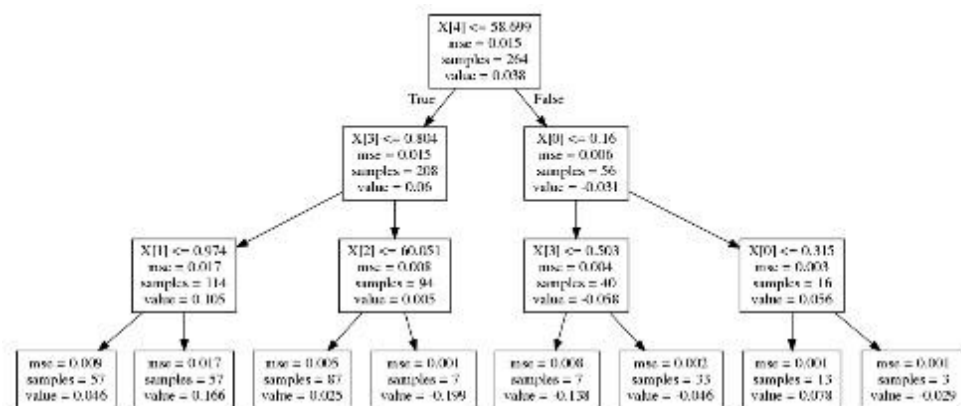
Data Science Professor





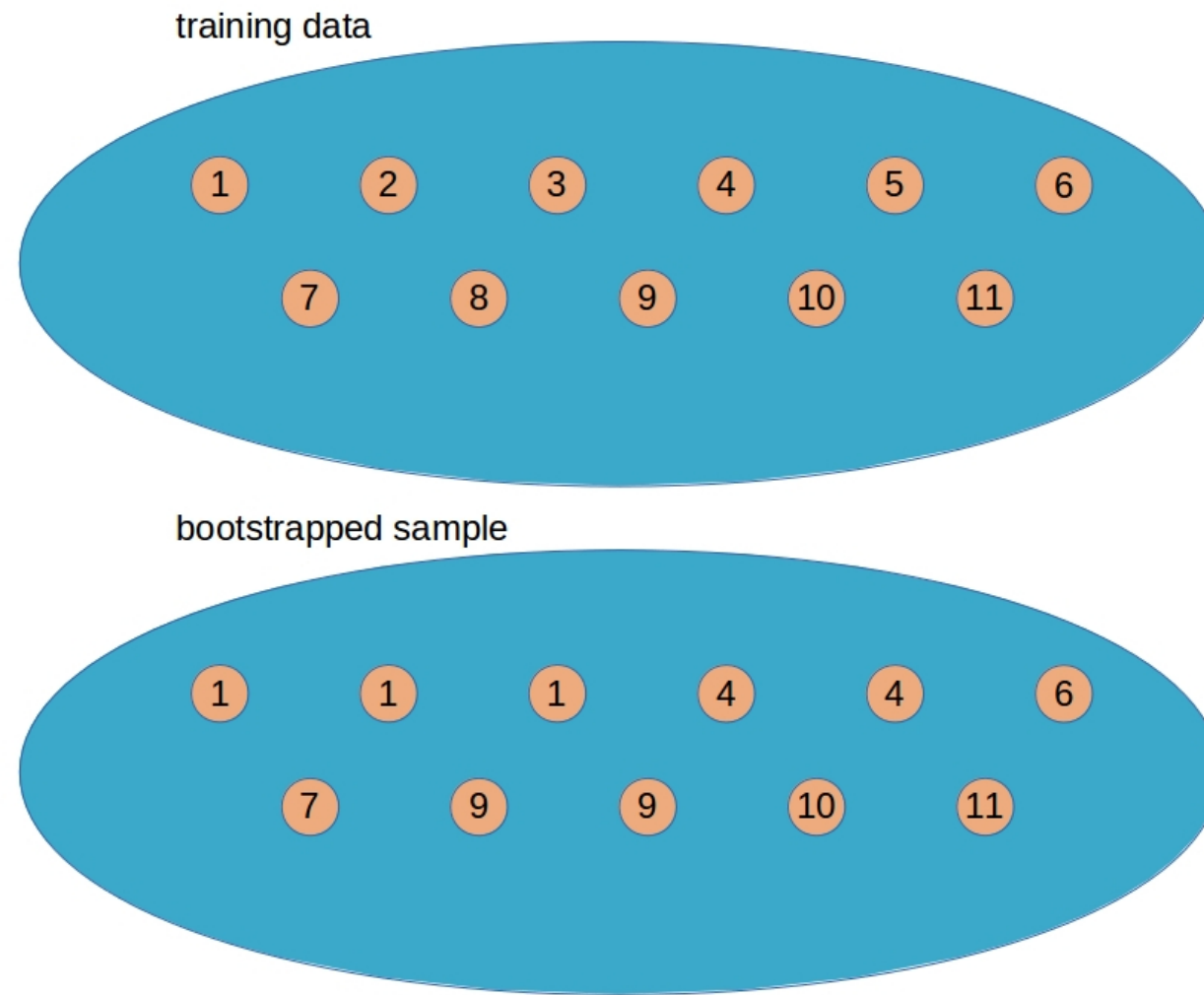


# Random forests





# Bootstrap aggregating (bagging)





# Feature sampling

## Random Forests

- A collection (ensemble) of decision trees
- Bootstrap aggregating (bagging)
- Sample of features at each split



# sklearn implementation

```
from sklearn.ensemble import RandomForestRegressor

random_forest = RandomForestRegressor()
random_forest.fit(train_features, train_targets)
print(random_forest.score(train_features, train_targets))
```





# Hyperparameters

```
random_forest = RandomForestRegressor(n_estimators=200,  
                                     max_depth=5,  
                                     max_features=4,  
                                     random_state=42)
```



# ParameterGrid

```
from sklearn.model_selection import ParameterGrid

grid = {'n_estimators': [200], 'max_depth': [3, 5], 'max_features': [4, 8]}

from pprint import pprint

pprint(list(ParameterGrid(grid)))

[{'max_depth': 3, 'max_features': 4, 'n_estimators': 200},
 {'max_depth': 3, 'max_features': 8, 'n_estimators': 200},
 {'max_depth': 5, 'max_features': 4, 'n_estimators': 200},
 {'max_depth': 5, 'max_features': 8, 'n_estimators': 200}]
```

# ParameterGrid

```
test_scores = []

# loop through the parameter grid, set hyperparameters, save the scores
for g in ParameterGrid(grid):
    rfr.set_params(**g) # ** is "unpacking" the dictionary
    rfr.fit(train_features, train_targets)
    test_scores.append(rfr.score(test_features, test_targets))

# find best hyperparameters from the test score and print
best_idx = np.argmax(test_scores)
print(test_scores[best_idx])
print(ParameterGrid(grid)[best_idx])
```

```
0.05594252725411142
{'max_depth': 5, 'max_features': 8, 'n_estimators': 200}
```



MACHINE LEARNING FOR FINANCE IN PYTHON

**Plant some random  
forests!**

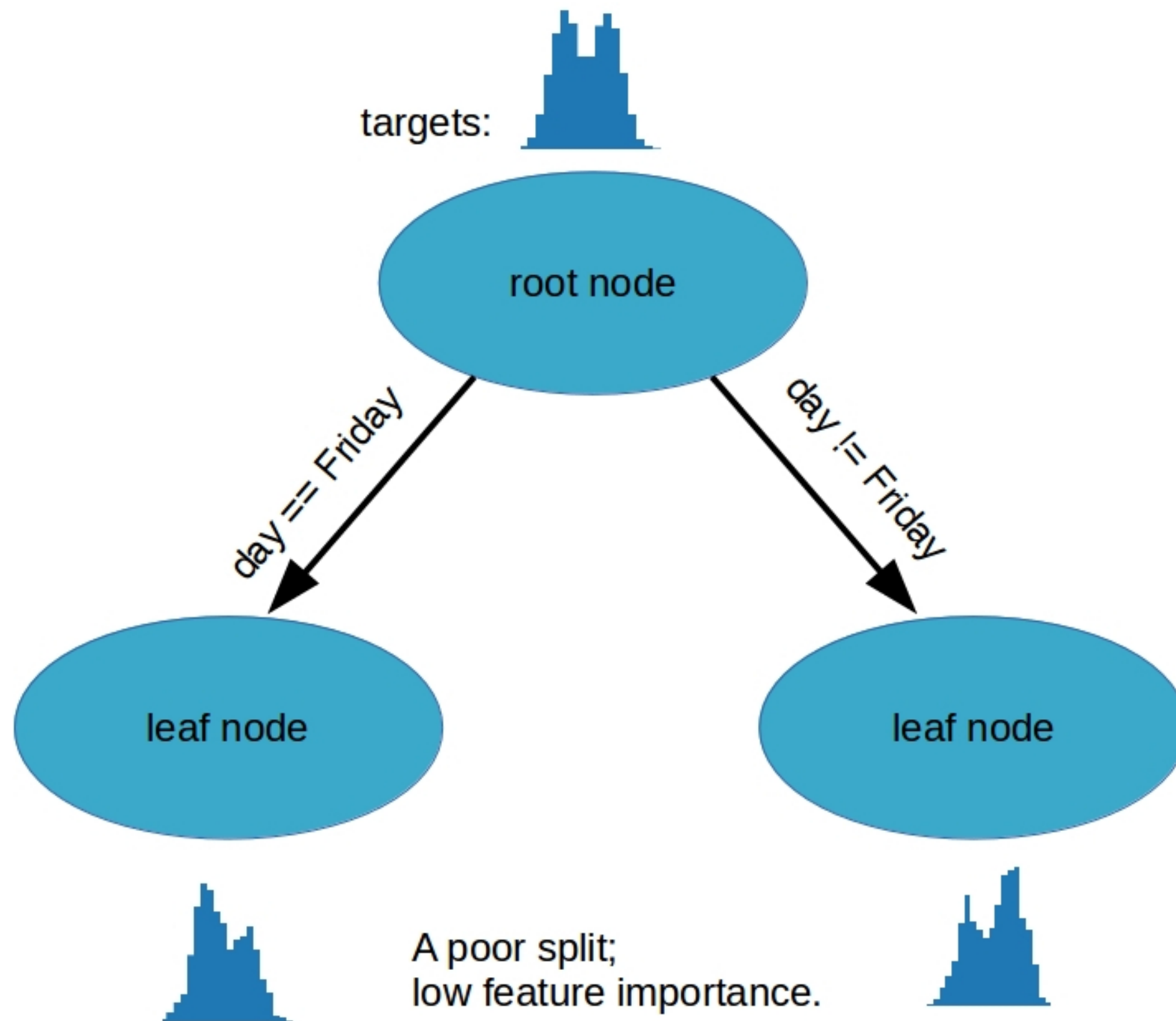


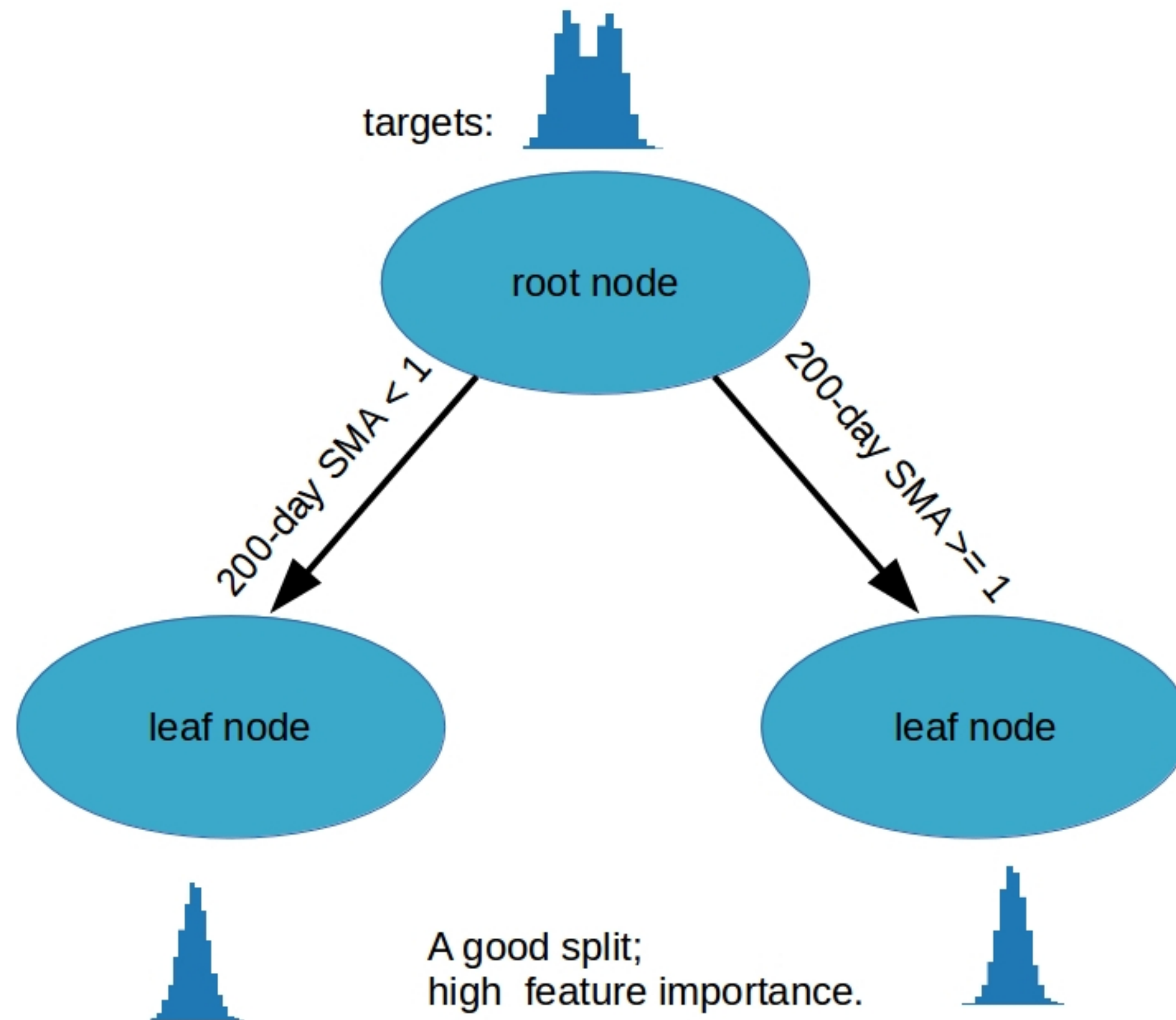
MACHINE LEARNING FOR FINANCE IN PYTHON

# Feature importances and gradient boosting

Nathan George

Data Science Professor







# Extracting feature importances

```
from sklearn.ensemble import RandomForestRegressor

random_forest = RandomForestRegressor()
random_forest.fit(train_features, train_targets)

feature_importances = random_forest.feature_importances_

print(feature_importances)
```

```
[0.07586547 0.10697602 0.12215955 0.23969227 0.29010304 0.0314028
 0.11977058 0.00276721 0.00246329 0.0026431  0.00615667]
```





# Sorting and plotting

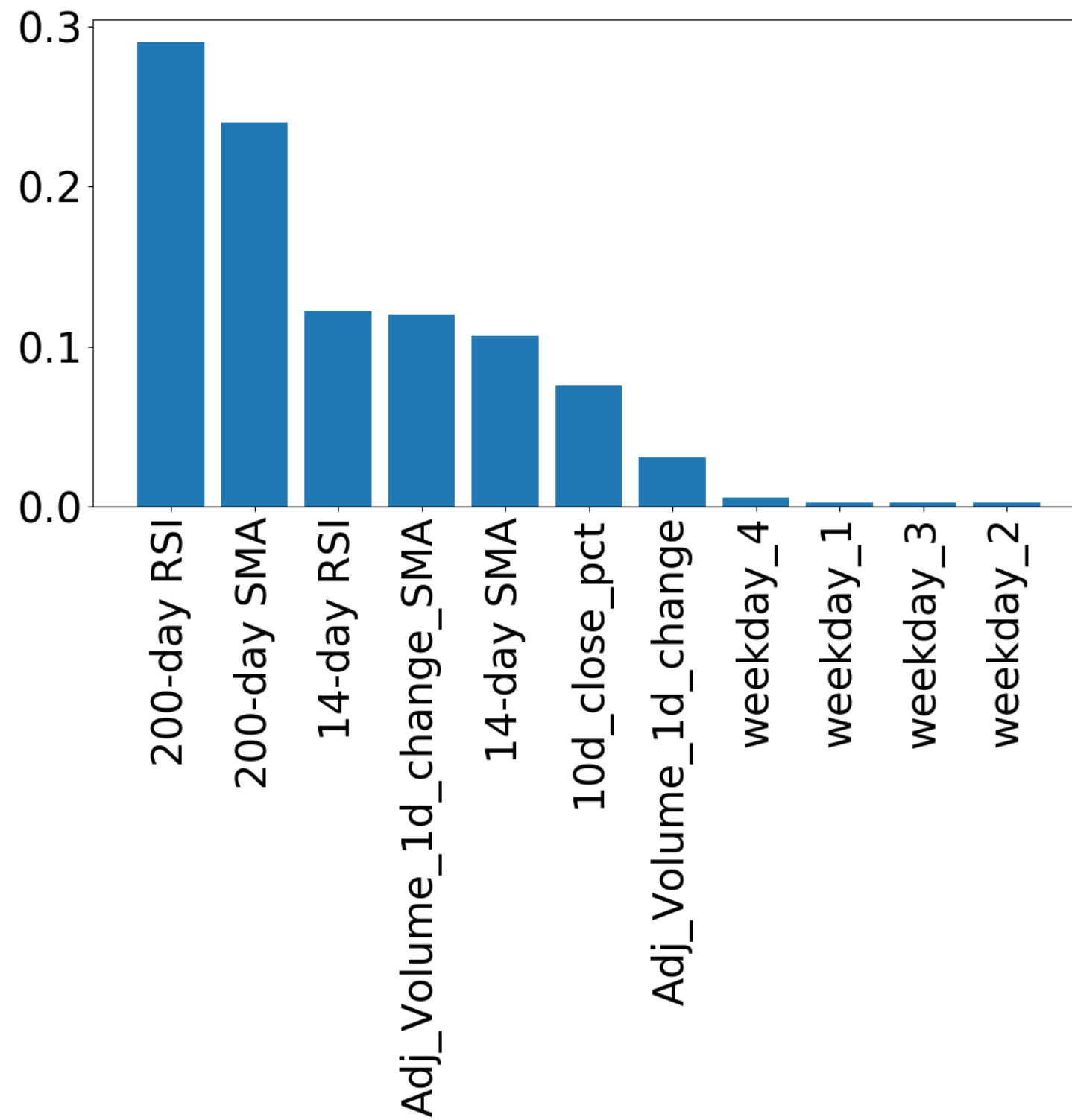
```
# feature importances from random forest model
importances = random_forest.feature_importances_

# index of greatest to least feature importances
sorted_index = np.argsort(importances)[::-1]

x = range(len(importances))
# create tick labels
labels = np.array(feature_names)[sorted_index]

plt.bar(x, importances[sorted_index], tick_label=labels)

# rotate tick labels to vertical
plt.xticks(rotation=90)
plt.show()
```

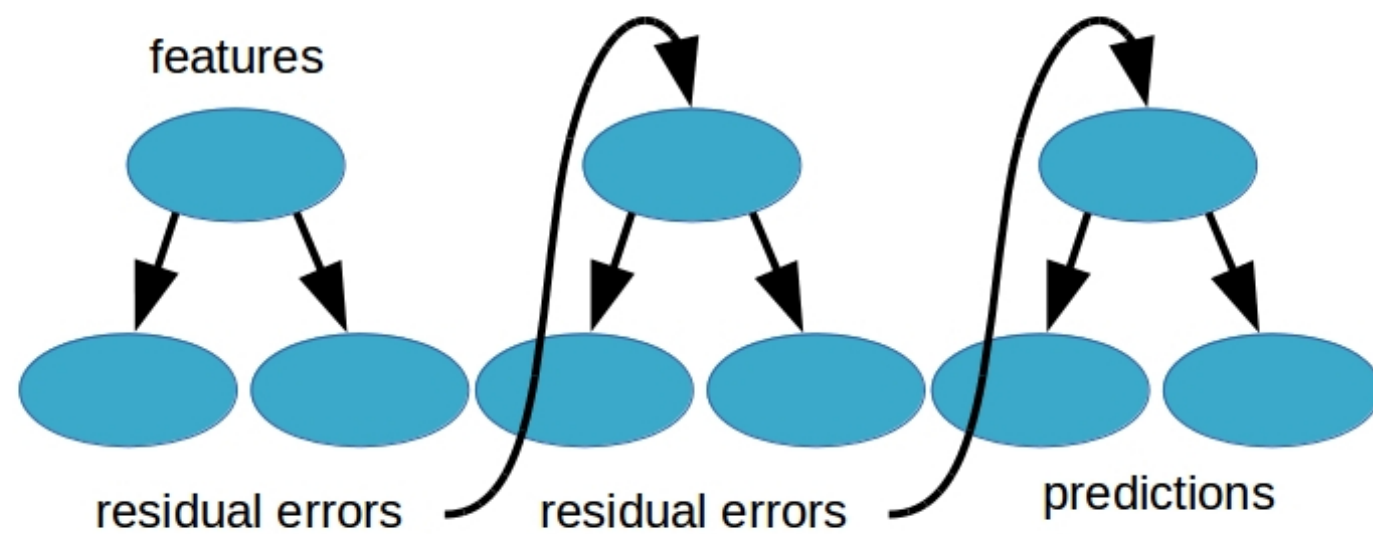




# Linear models vs gradient boosting



<http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting/>





# Boosted models

Available boosted models:

- Gradient boosting
- Adaboost



# Fitting a gradient boosting model

```
from sklearn.ensemble import GradientBoostingRegressor

gbr = GradientBoostingRegressor(max_features=4,
                                learning_rate=0.01,
                                n_estimators=200,
                                subsample=0.6,
                                random_state=42)

gbr.fit(train_features, train_targets)
```



## MACHINE LEARNING FOR FINANCE IN PYTHON

**Get boosted!**