Introduction to Machine Learning Course

Jae Yoon Jeong, Jaewook Lee, Taeseok Kang

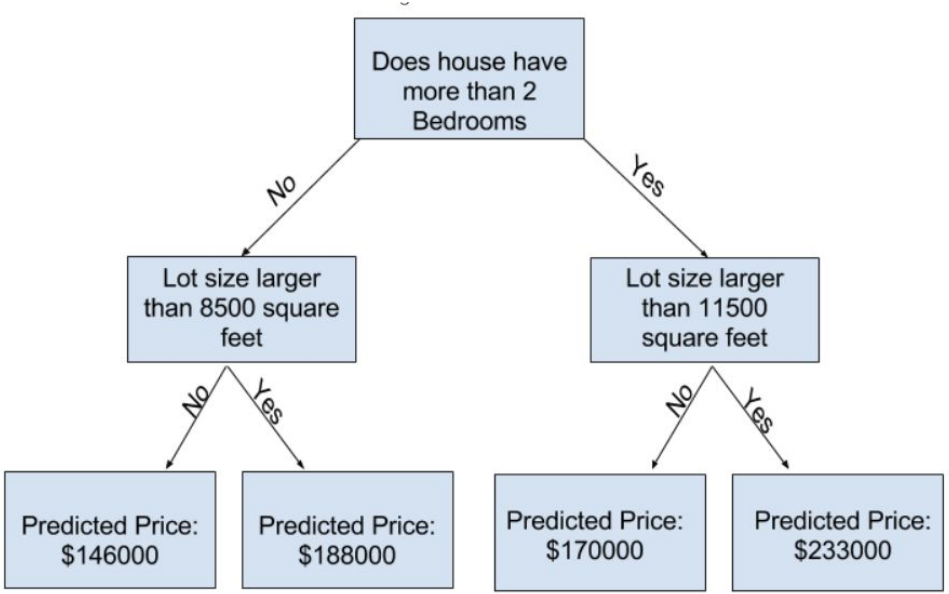
Learn the core ideas of ML, build our own model

Topic: Basics of Machine Learning

Duration: 2 Weeks (2 Weeks from previous semester)

How Models Work

- Prediction of housing prices in Iowa
- Decision Tree
- Examining data from a given set to create Decision Tree Models
- Analyzing patterns from data: Fitting & Modeling



Basic Data Exploration

- The use of pandas library and how to prepare data for analysis
- Use of read_csv() and displaying the data using functions such as describe()
- Missing values (not too in-depth)

```
import pandas as pd

# Path of the file to read
iowa_file_path = '../input/home-data-for-ml-course/train.csv'

# Fill in the line below to read the file into a variable home_data
home_data = pd.read_csv(iowa_file_path)

# Call line below with no argument to check that you've loaded the data correctly
step_1.check()
```

```
# Print summary statistics in next line
home_data.describe()
```

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1
count	1460.000000	1460.000000	1201.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1452.000000	1460.000000
mean	730.500000	56.897260	70.049958	10516.828082	6.099315	5.575342	1971.267808	1984.865753	103.685262	443.639726
std	421.610009	42.300571	24.284752	9981.264932	1.382997	1.112799	30.202904	20.645407	181.066207	456.098091
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1872.000000	1950.000000	0.000000	0.000000
25%	365.750000	20.000000	59.000000	7553.500000	5.000000	5.000000	1954.000000	1967.000000	0.000000	0.000000
50%	730.500000	50.000000	69.000000	9478.500000	6.000000	5.000000	1973.000000	1994.000000	0.000000	383.500000
75%	1095.250000	70.000000	80.000000	11601.500000	7.000000	6.000000	2000.000000	2004.000000	166.000000	712.250000
max	1460.000000	190.000000	313.000000	215245.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000

Machine Learning Model

- Selecting data for Modeling
  - Choosing the correct dataset
  - Dataset we use to create our model is called “features”
- Features (x)
- Prediction target (y)
- Scikit-Learn library to create our model
  - Define, Fit, Predict, Evaluate

```
melbourne_features = ['Rooms', 'Bathroom', 'Landsize', 'Lattitude', 'Longtitude']
```

```
from sklearn.tree import DecisionTreeRegressor
#specify the model.
#For model reproducibility, set a numeric value for random_state when specifying the model
iowa_model = DecisionTreeRegressor(random_state = 13)

# Fit the model
iowa_model.fit(X,y)
```

```
predictions = iowa_model.predict(X)
print(predictions)

# Check your answer
step_4.check()
```

```
[208500. 181500. 223500. ... 266500. 142125. 147500.]
```

Model Validation

- Evaluation process of our created model
- Measuring the predictive accuracy of our model
- Separation of validation data and training data
- Comparison of the mean absolute error

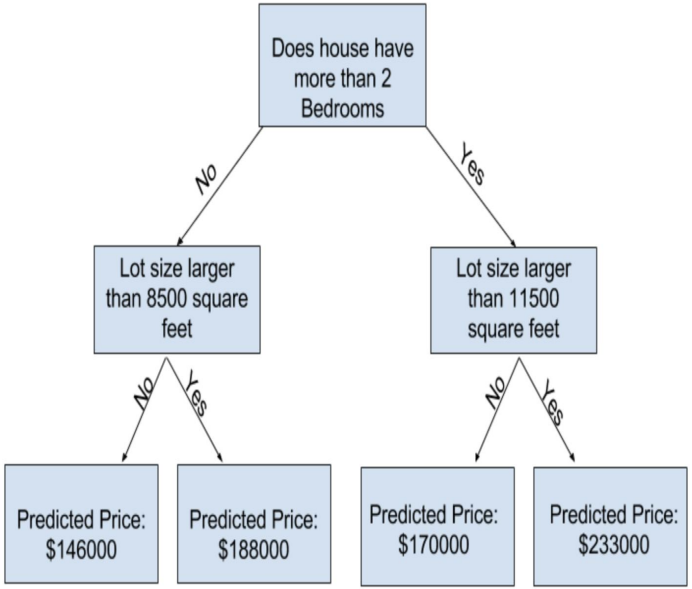
```
# Predict with all validation observations
val_predictions = iowa_model.predict(val_X)
```

```
from sklearn.metrics import mean_absolute_error
val_mae = mean_absolute_error(val_y, val_predictions)

# uncomment following line to see the validation_mae
print(val_mae)

# Check your answer
step_4.check()
```

```
29652.931506849316
```

Underfitting and Overfitting

- Determining the accuracy of the model by checking underfitting and overfitting
- The tree depth is how many splits the decision tree has
- If the tree has too many splits, it is called overfitting *(unreliable due to too many specifications)*
- If the tree has too few splits, it is called underfitting *(unreliable due to failure in capturing distinctions and patterns in the data)*



- Ideally, we would want our model to be in the middle ground for the most accurate results

```
def get_mae(max_leaf_nodes, train_X, val_X, train_y, val_y):
    model = DecisionTreeRegressor(max_leaf_nodes=max_leaf_nodes, random_state=0)
    model.fit(train_X, train_y)
    preds_val = model.predict(val_X)
    mae = mean_absolute_error(val_y, preds_val)
    return(mae)
```

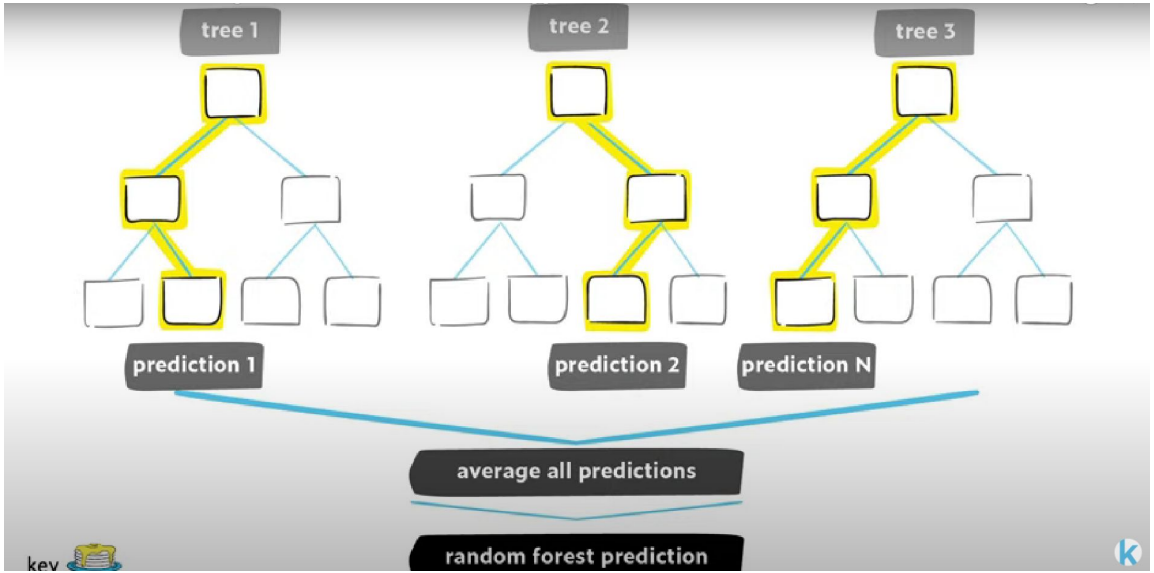
```
candidate_max_leaf_nodes = [5, 25, 50, 100, 250, 500]
# Write loop to find the ideal tree size from candidate_max_leaf_nodes


# Store the best value of max_leaf_nodes (it will be either 5, 25, 50, 100, 250 or 500)
scores = {leaf_size: get_mae(leaf_size, train_X, val_X, train_y, val_y) for leaf_size in candidate_max_leaf_nodes}
best_tree_size = min(scores, key=scores.get)
```

```
print(best_tree_size)
```

```
100
```

```
final_model = DecisionTreeRegressor(max_leaf_nodes=best_tree_size, random_state=1)
final_model.fit(X, y)
```

Random Forests



- Built up of many individual decision trees
- Makes the prediction by averaging of each each component trees
- Much more accurate than single decision tree

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error

forest_model = RandomForestRegressor(random_state=1)
forest_model.fit(train_X, train_y)
melb_preds = forest_model.predict(val_X)
print(mean_absolute_error(val_y, melb_preds))
```

191669.7536453626

29652.931506849316

Thank You

Any questions?