17. Random Forest :-

MINDLESS DATE: _____

- Random forest have the ability to greatly increase the performance based on expanding ideas from the Decision Tree.
- Random Forests are known as ensemble lerners, since they rely on an ensemble of models (multiple decision tree)

Section Overview
- Motivation and History
- Hyperparameter
- Random forest classification
- Random forest Regression
  - Comparing many regression models (SVR, DTR, etc)

02 Random Forest

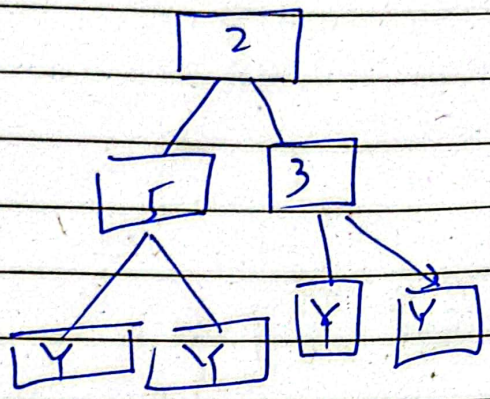
Theory & Intuition

Question we can ask
- Why not just use Decision Tree?
- Motivation behind ~~decision tree~~ Random Forest and how they improve decision tree.
-

M T W T F S DATE:

- Imagine a dataset

1	2	3	4	5	Y



we will always use certain feature every time we try to split mean we will have same node in decision tree unless we change hyperparameter so some features might be neglected.

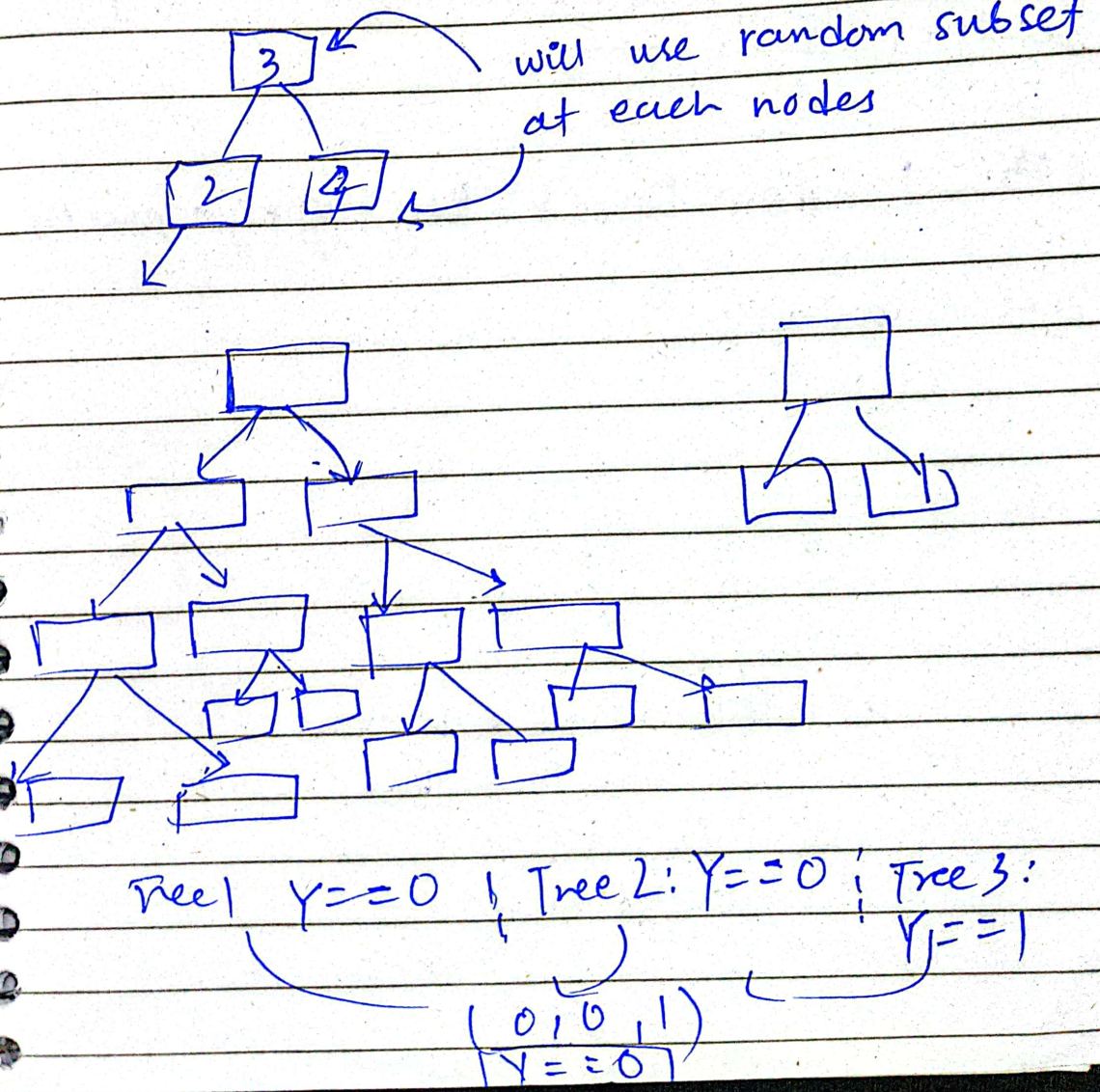
Decision Tree

- No guarntee of using all features
- Root node will always be same. (using same rule)
- Root node will have a huge influence on tree.
- We could try adjusting rules, such as:
  - splitting criterion (Information Gain)
  - Minimum Gini Impurity Decrease
  - Setting Depth limits
  - limit on number of terminal leaf nodes.
- However even will all these added hyperparameter adjustments the single decision tree is still limited:

- Single feautrue for root node
- splitting criteria can lead to some features not being used.
- potential for overfitting the data

- 1995: Tim Kam Ho present Random Decision Forest at 3rd International Conference on Document Analysis and Recognition in Montreal.

↳ create subsets of randomly picked features at each potential split



M T W T F S

DATE:

We can also see probablity of the classification.

- Ho and other reasercher shows that adding the random features subspace constraint could allow trees to grow much deeper without causing overfitting.
- Mathimatical explanation of this was shown by Eugene Kleinberg's theory of stochastic discrimination.
- 2001 : Leo Breiman published Random Forests in the journal Machine learning.
- Introduces bootstrapping samples and out of bag error.

003 Random Forest - Key Hyperparameters

```
DecisionTreeClassifier(*, criterion='gini',

                       )

RandomForestClassifier( n_estimators,
                        max_features='auto',
                        bootstrap=True,
                        oob_score=False
                       )
```

NOTES DATE:

Random Forest Hyperparameter different from:

- Number of Estimators → How many decision tree used
- Number of features → How many features to include in each subset
- Bootstrap sample
- Out_of-Bag Error

→ Allow for bootstrap sampling of each training subset of features?

→ Calculate OOB error during training?

004 Random Forests Hyperparameters Estimater and Features

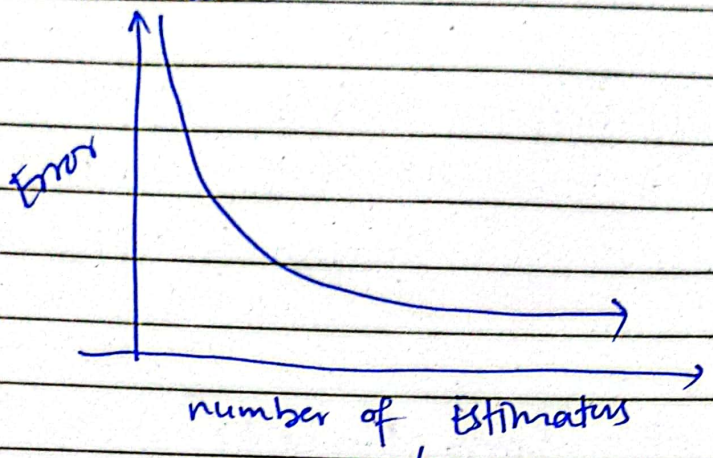
- Number of Estimators:
  - Intuitively, we know the more decision tree, the more opportunities to learn from variety of features subset combinations.
  - Is there a limit to adding more trees?
  - Is there a danger of overfitting?

- How to choose number of trees?
  - Reasonable Default values: 100
  - publication suggest 64-128 trees.
  - cross Validate a grid search of trees.

M T W T F S DATE:

- plot Error versus number of trees (similar to elbow method of KNN)
  - Should notice diminishing error reduction after some N trees.

Error vs Trees:



- After a certain number of trees, two things that can occur:
  - Different random selection don't reveal any more information
    - Trees become highly correctly
  - Different random selections are simply duplicating trees that have already been created.

- This allows us to be quite lenient in setting the number of estimators hyperparameters, as overfitting is of minimal concern.

NOTES DATE:

* Now lets us discuss the how to choose number of features to randomly select at each split.

- Number of Feature
  - how many features to include in each subset when splitting at a node?
  - original publication suggested subset of $\log_2(N+1)$ random features in subset given a set of N total features.

- From Leo Breiman's offical page on Random Forests:
  - An Interesting difference between regression and classification is that the correlation increase quite slowly as the number of features used increase,".

→ current suggested convention is sqrt(N) in the subset given N features

→ Later suggested by Breiman indated N/3 may be more suitable for regression task, typically tasks

M T W T F S DATE:

than sqrt(N).

- ISLR indicates these to be as tuning parameter with SQRT(N) as a good starting point.
  - All previous were shown using empirical method.

- Start with 100 as default estimated and then Grid search

- Start with sqrt(N), grid search for other possible values (N/3)

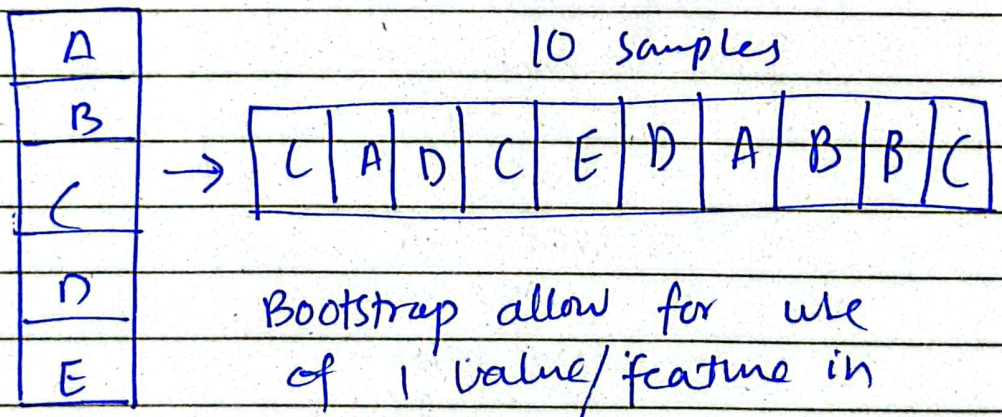
005 Random Forest - Bootstraping and out-of-Bdg Error

NOTES DATE:

- Random Forest Hyperparameters:
  - Bootstrap samples
    Allow for bootstrapi sampling of each training subset of features?
  - Out-of-Bag Error
    - calculate OBB error during training ?

What is Bootstrapping ?

- A term used to describe "random sampling with replacement".
- Let see a quick example of given a set of letters.



Bootstrap allow for use of 1 value/feature in multiple places

default = True

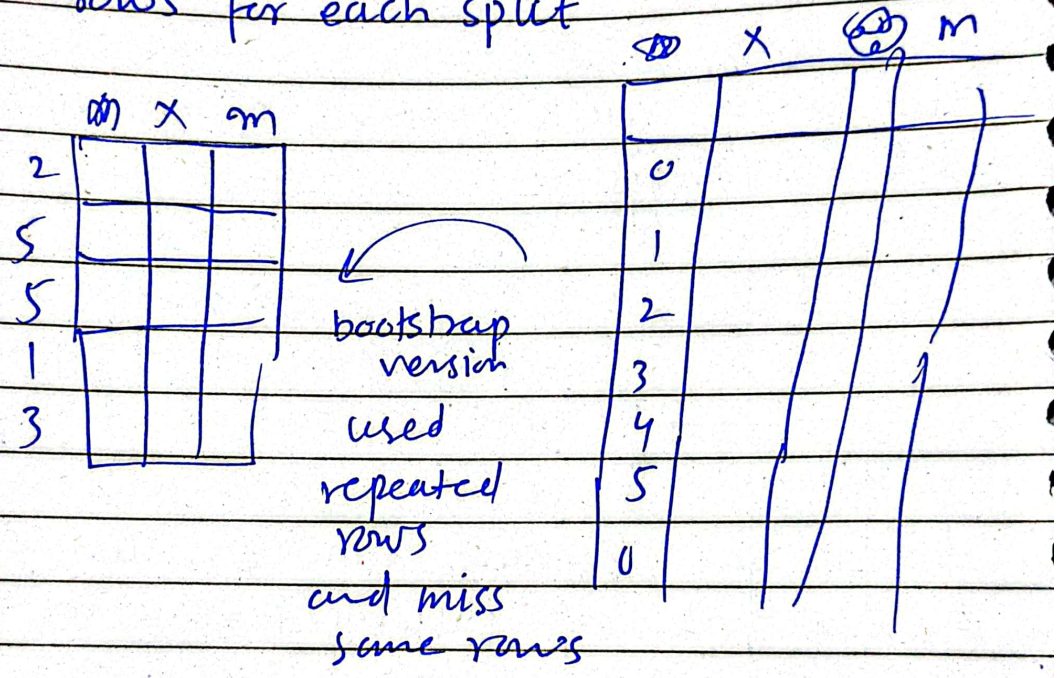
Bootstraping is using some rows multiple time and not using some row in Decision tree to have more non correlation between the decision trees,

M T W T F S DATE:

Bootstrapping in Random Forest:

- Recall for each split we are randomly selecting a subset of features.
- This randomly subset of features help create more diverse trees that are not not correlated.

- To further differentiate trees, we could bootstrap a selection of rows for each split



- Bootstrap can be set to False during training (it is True by default)
- Boostrapiy is yet another hyperparameter meant to reduce correlation between trees, since trees are then trained on different subsets of features columns and data Rows.

NOTES DATE:

Calculate out-of-bag error refering to bagging

What is Bagging?

Recall to ~~calculate~~ actually use a Random forest, we use bootstrapped data and then calculate a prediction based on the ~~aggregated~~ (average) prediction of the trees:

- Classification: Most Voted Y class
- Regression: Average predicted Ys

- ~~If we performed~~ bootstrapping when building out trees, this mean that for certain trees, certain rows of data were not used in training. those not used rows are called out-of-bag samples.

- Not used for constructing some trees.
- We could use these to get performance test metrics on trees that did not use these rows!

then we can use that out-of-bag sample to find OOB error on each Decision tree.

M T W T F S DATE:

- Note the OOB score doesn't effect training its just calculate OBB error when bootstrap is true mean. setting OBB to true doesn't change Decision Trees but setting bootstrapping to True does change the structure of decision trees.

- Note that OBB score is also limited to not using all the trees in the random forest, it can only be calculated on trees that did not use the OBB data.
- Due to not using of the entire random forest, the default value of OBB score hyperparameters is set to false.

(Random Forest Classifier)

008 - Coding Classification - RFC

Part 1

```
df = pd.read_csv("penguin")
df = df.dropna()
df.head()

X = pd.get_dummies(df.drop("species", axis,
                           drop_first=True))
Y = df['species']

train_test_split(X, y, test_size=0.3, ran
                                    =101)
```

M I D W I F E S DATE:

```
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n_estimators=100,
                             max_features={'auto',
                                           'sqrt',
                                           "log2"}
                             random_state=101)
```

int = max

↳ float = fraction

↑ int(max_features * n_features)

max features

auto = sqrt(N)

sqrt = sqrt(N)

$\log_2 N = 5 \log_2(N)$

```
rfc.fit(X_train, y_train)

preds = rfc.predict(X_test)
# predics
plot_confusion_matrix(model, X_test, y_test)
print(classification_report(y_test, y_preds)
```

007 Coding classification with Random Forest - part two

M T W T F S DATE:

```
df = pd.read_csv("../data_banknote.csv")
df.head()

sns.pairplot(df, hue = "class")

X = df.drop('class', axis = 1)
y = df['class']

from sklearn
train_test_split(X, y, test_size = 0.15,
                 random_state = 101

from sklearn.model_selection import GridSearchCV

from sklearn.
n_estimators = [64, 100, 128, 200]
max_features = [2, 3, 4]
bootstrap = [True, False]
oob_score = [True, False],

params = grid = {

}

rfc = RandomForestClassifier()
grid = GridSearchCV(rfc, param_grid)

grid.best_params_
```

MIDIES DATE:

```
rfc = RandomForestClassifier(max_features=2,
                             n_estimators=200,
                             oob_score=True)
rfc.fit(X_train, y_train)
rfc.oob_score  → True or False
rfc.oob_score_

predict = rfc.predict(X_test)
classification_report( )
confusion_matrix( )

error = [ ]
misclassification = [ ]
for n in range(1, 200):
    rfc = RandomForestClassifier(n_ )
    rfc.fit(X_train, y_train)
    err = 1 - accuracy_score(y_test, y_preds)
    n_missed = np.sum(preds != y_test)
    errors.append(err)
    misclassification.appends(n_missed)

plt.plot(range(1, 200), error)
```

08 Coding Regression with Random Forest - part one

M T W T F S DATE:

Discussion about Dataset

9 Coding Regression with part 2

```
df = pd.read_csv("rock_reboud")
df.columns = ['signal', 'density']
sns.scatterplot(x='signal', y='density', data=df)

x = df['signal'].values.reshape(-1,1)   error if not done
y = df['density']

x_train, y_test, u
lr = model = LinearRegression()
lr_model.fit(X_train, y_train)
lr_preds = lr_model.predict(X_test)
mean_absolute_error(y_test, lr_preds)

signal_range = np.arange(0, 100)
signal_pred = lr_model.predict(signal_range.reshape(-1,1))
```

10 Regression - part 2

MODULES DATE:

```
def run_model(model, X_train, y_train, X_test, y_test)
    model.fit(X_train, y_train)

    preds = model.predict(X_test)
    rmse = np.sqrt(mean_sq(y_test, preds))
    mae =
    print(f'MAE: {mae}')
    print(f'(RMSE{ }')

# plot Result
    Signal range = np.arange(0, 1000)
    sigpred = model.predict(sig.reshape
                            (-1, 1

    plt.figure( )
    sns.scatterplot(X_ )
    plt.plot

from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import polynomialFe

pipe = make_pipeline(polynomialfeature(degree=2)
                     LinearRegression( ))
run_model(pipe, X_train, y_train, X_test, y_test)
```

11 - Loading Regression with Random Forest Regressor

M T W T F S DATE:

```
#
from sklearn.neighbors import KNeighborsRegressor

k_values = [1, 5, 10]
for n in k_values:
    model = KNeighborsModel(n_neighbors = n)
    run_model(model, X_train, y_train, X_test)
```

model = DecisionTree

SVR from sklearn.svm import SVR

```
svr = SVR()
params_grids = {'C': [0.01, 0.1, 1, 5, 10, 100, 1000],
                'gamma': ['auto', 'scale']}

grid = GridSearchCV(SVR, params_grids
```

```
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor(n_estimators = 10)
```