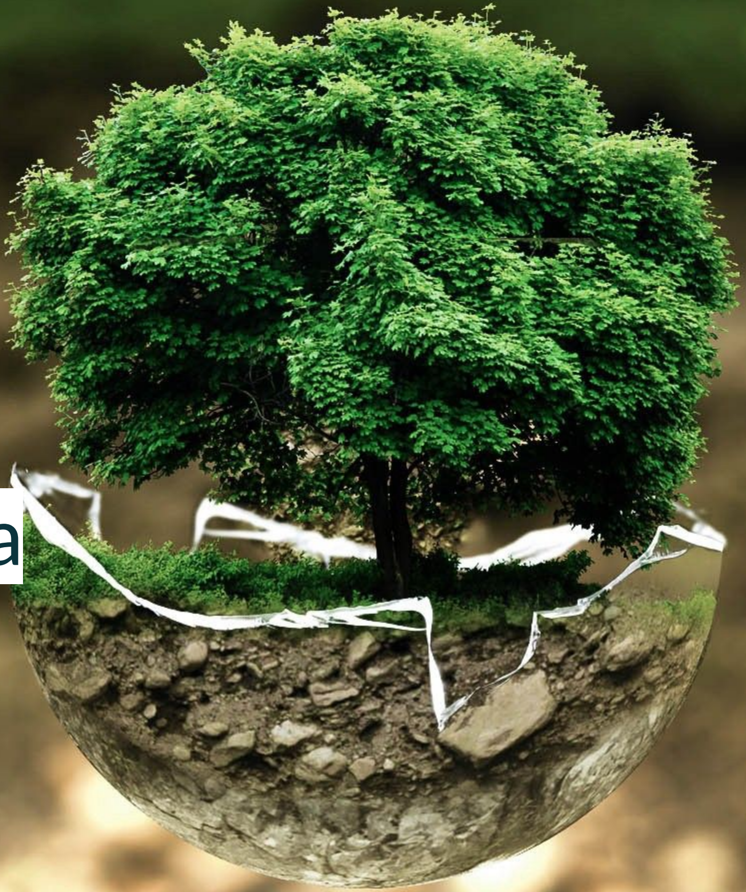
Lesson #17
Decision Trees

notas

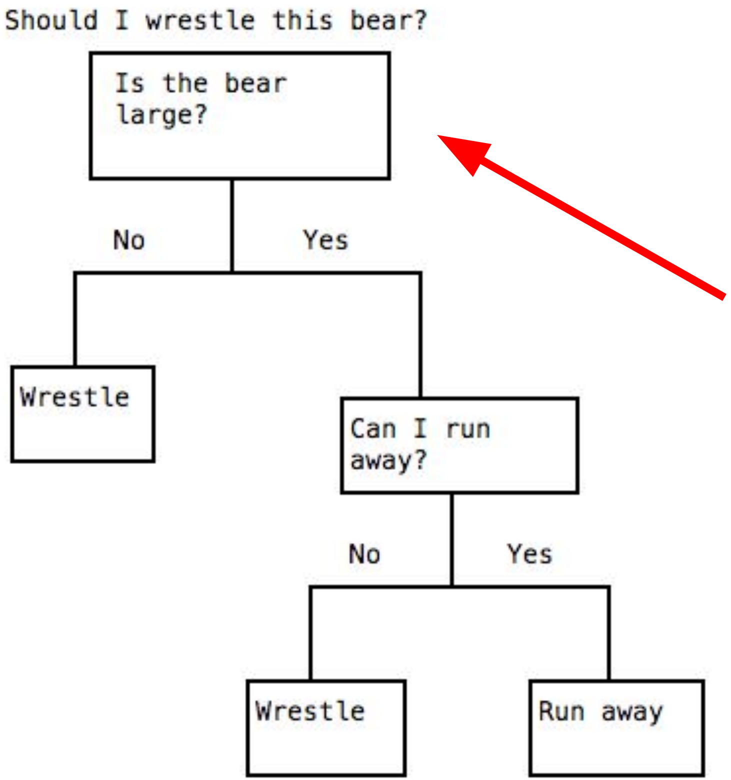
- Introduction to Decision Tree
- Converting categorical variables
- Splitting Data
- Decision Trees as flows of data
- Entropy & Gini
- Information gain
- Applying Decision Trees
- Overfitting problem
- Case study: classification problem

notas

How can an algorithm be represented as a tree?

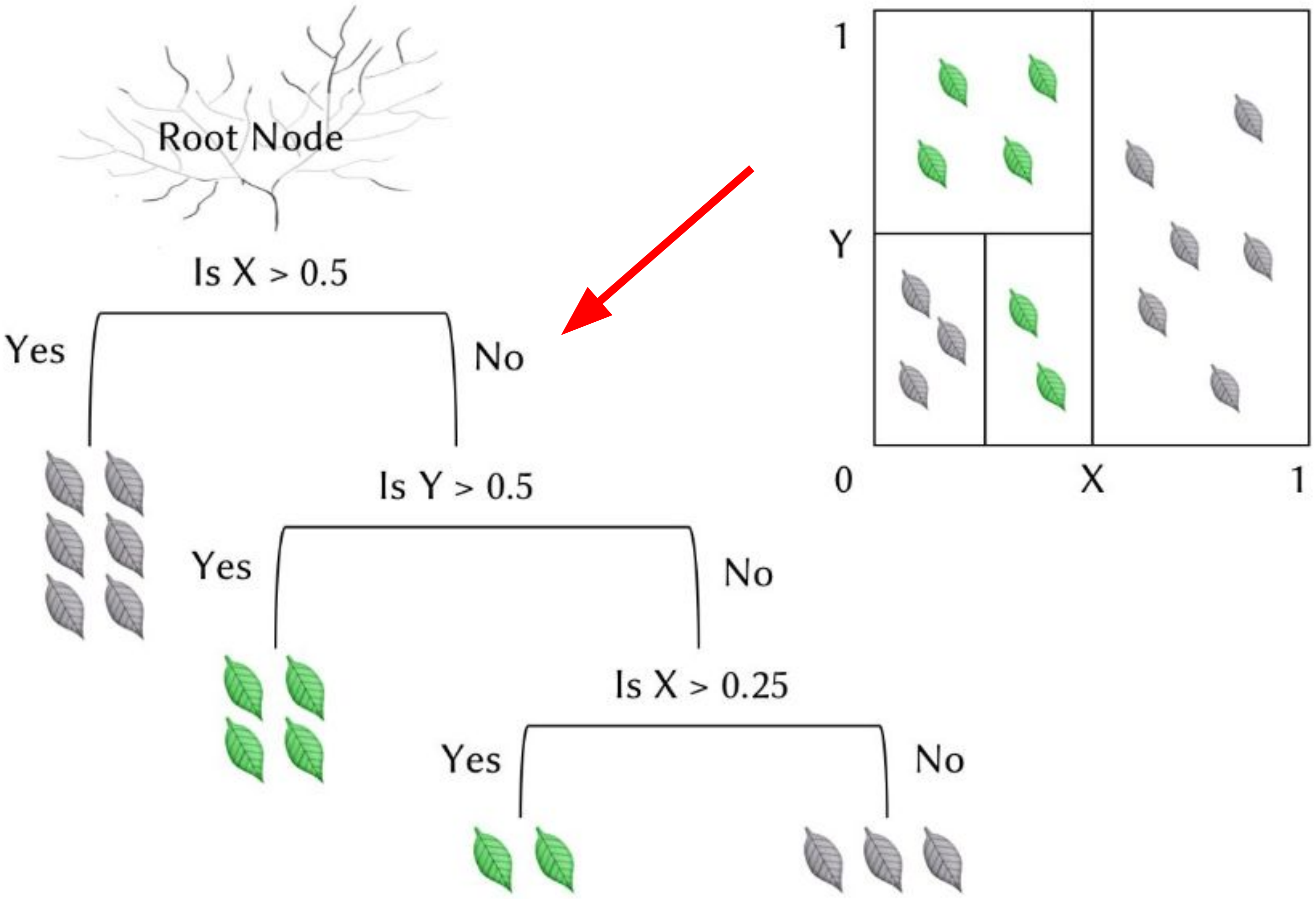


Decision Tree (classification)



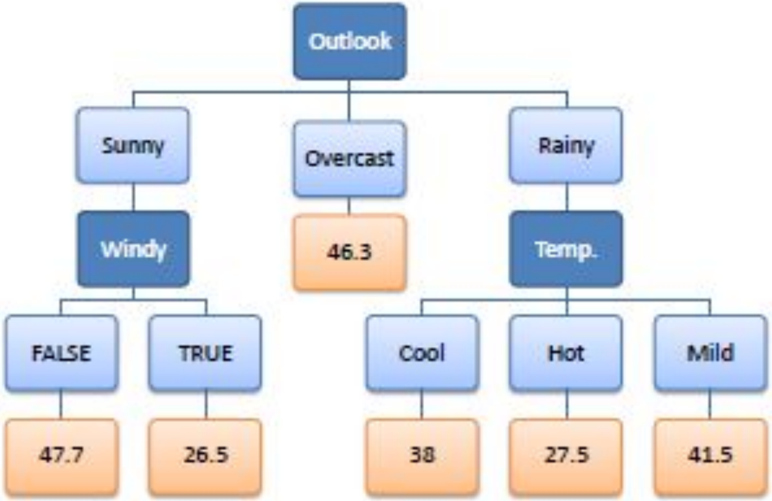
Bear name	Size	Escape possible?	Action
Yogi	Small	No	Wrestle
Winnie	Small	Yes	Wrestle
Baloo	Large	Yes	Run away
Gentle Ben	Large	No	Wrestle

Decision Tree (classification)



Decision Tree (regression)

Predictors				Target
Outlook	Temp.	Humidity	Windy	Hours Played
Rainy	Hot	High	False	25
Rainy	Hot	High	True	30
Overcast	Hot	High	False	46
Sunny	Mild	High	False	45
Sunny	Cool	Normal	False	52
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	35
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	46
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	52
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30




CASE STUDY

Adult Data Set

Download: Data Folder, Data Set Description

Abstract: Predict whether income exceeds $50K/yr based on census data. Also known as "Census Income" dataset.

Data Set Characteristics:	Multivariate	**Number of Instances:**	48842	**Area:**	Social
Attribute Characteristics:	Categorical, Integer	**Number of Attributes:**	14	**Date Donated**	1996-05-01
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	1564397

high_income
>50K
<=50K
<=50K
>50K
<=50K

Source:

Donor:

Ronny Kohavi and Barry Becker
Data Mining and Visualization
Silicon Graphics.
e-mail: ronnyk '@' live.com for questions.

Data Set Information:

Extraction was done by Barry Becker from the 1994 Census database. A set of reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1)&& (HRSWK>0))

Prediction task is to determine whether a person makes over 50K a year.

Dataset info

Number of variables	15
Number of observations	32561
Total Missing (%)	0.0%
Total size in memory	3.7 MiB
Average record size in memory	120.0 B

Variables types

Numeric	6
Categorical	9
Boolean	0
Date	0
Text (Unique)	0
Rejected	0
Unsupported	0

Warnings

- capital_gain has 29849 / 91.7% zeros Zeros
- capital_loss has 31042 / 95.3% zeros Zeros
- Dataset has 24 duplicate rows Warning

Be careful

Converting Categorical Variables

```
income.workclass.head()
```

```
0              State-gov
1       Self-emp-not-inc
2                Private
3                Private
4                Private
```

```
# Convert a single column from text categories to numbers
col = pd.Categorical(income["workclass"])
income["workclass"] = col.codes
income.workclass.head()
```

```
0     7
1     6
2     4
3     4
4     4
```

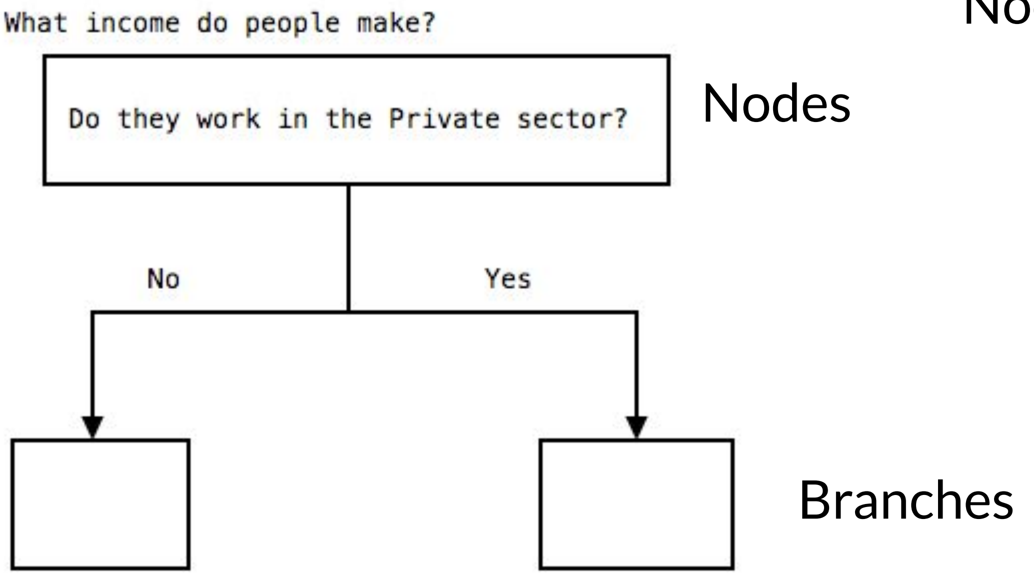
```
col.categories[7]
```

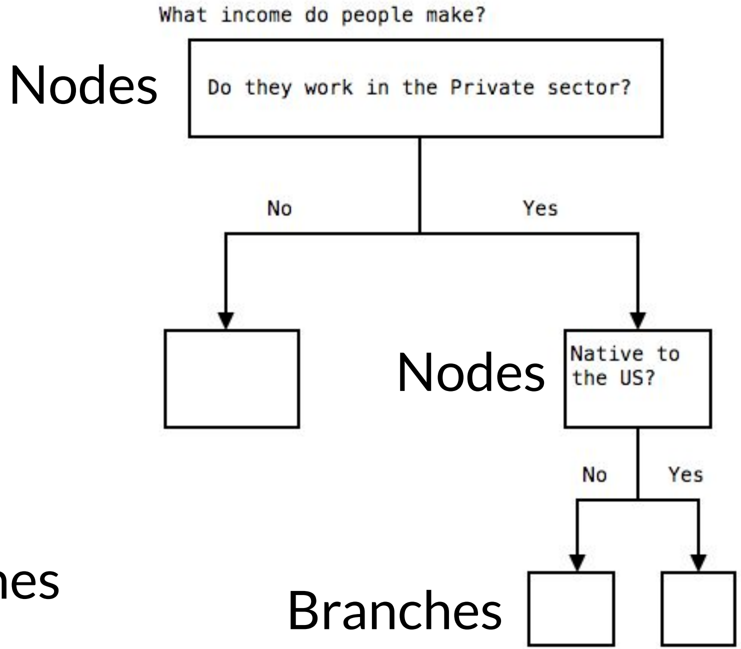
```
' State-gov'
```

```
col.categories
```

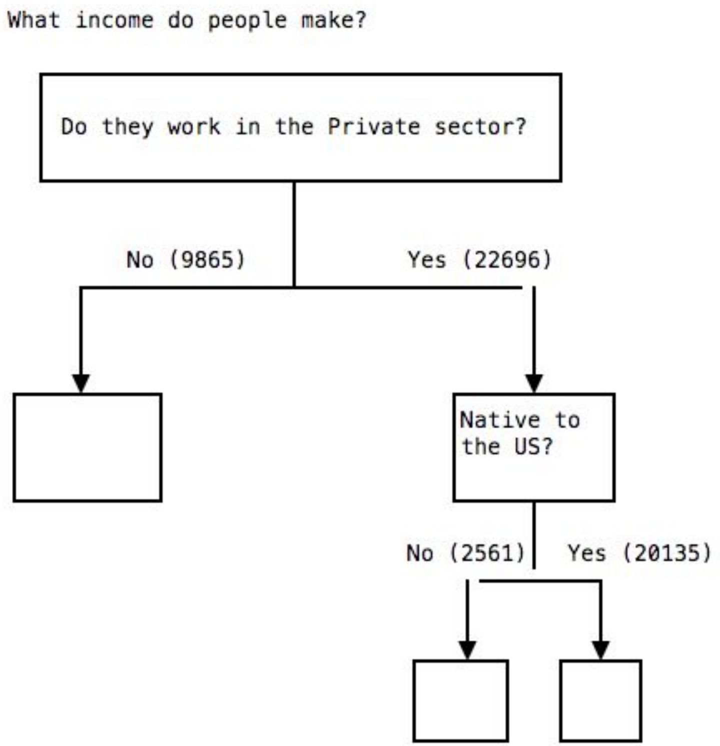
```
Index([' ?', ' Federal-gov', ' Local-gov', ' Never-worked', ' Private',
       ' Self-emp-inc', ' Self-emp-not-inc', ' State-gov', ' Without-pay'],
      dtype='object')
```

Splitting Data





Decision Tree as Flows of Data



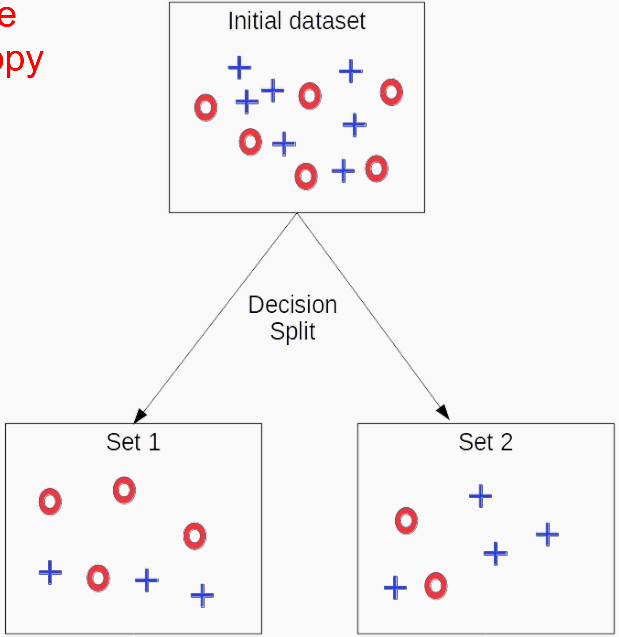
We'll need to continue splitting nodes until we get to a point where all of the rows in a node have the same value for high_income.

Entropy

Entropy is an indicator of how messy your data is.

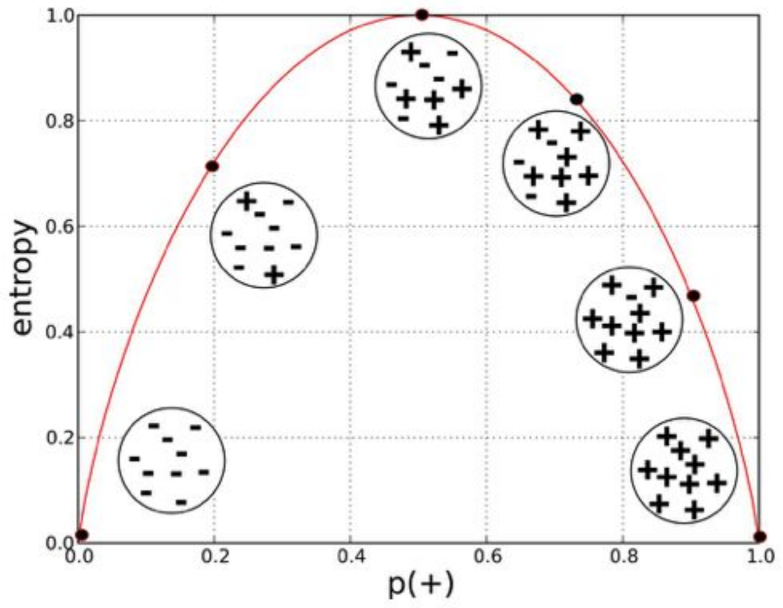
Why Entropy in Decision Trees?

Decrease
the entropy



- The goal is to tidy the data.
- You try to separate your data and group the samples together in the classes they belong to.
- You maximize the purity of the groups as much as possible each time you create a new node of the tree
- Of course, at the end of the tree, you want to have a clear answer.

Mathematical definition of entropy



- Suppose a set of S items, these items fall into two categories:
  - +: n items
  - -: m items
  - S = n+m
- p = n/S, q =m/S
- p + q = 1
- E = -p log(p) - q log(q)

Generalization

Feature X

$$E(x) = -\sum_{i=1}^{c} P(x_i) \log_b P(x_i)$$

$P(x_i)$ is the fraction of examples in a given class i

c = pd.AA.unique()

Entropy using the frequency table of one attribute

```
high_income
1
1
0
0
1
```

$$E(x) = -\sum_{i=1}^{c} P(x_i)\log_b P(x_i)$$

$$E(high\_income) = -\left(\frac{2}{5} \times log_2 \frac{2}{5}\right) - \left(\frac{3}{5} \times log_2 \frac{3}{5}\right)$$

$$E(high\_income) = 0.53 + 0.44$$

$$E(high\_income) = 0.97$$

Entropy using the frequency table of two attributes

age	high_income	split_age
25	1	0
50	1	0
30	0	0
50	0	0
80	1	1

$\frac{4}{5}$

$\frac{1}{5}$

split_age is based on median of age (suppose equal to 50)

$$E(T,X) = \sum_{c \in X} \frac{|X_c|}{|X|} E(T|X_c)$$

$$c_0 = \frac{4}{5} \times E([1,1,0,0])$$

$$c_1 = \frac{1}{5} \times E([1])$$

$$E(T,X) = c_0 + c_1$$

$$= 0.17$$

Information Gain

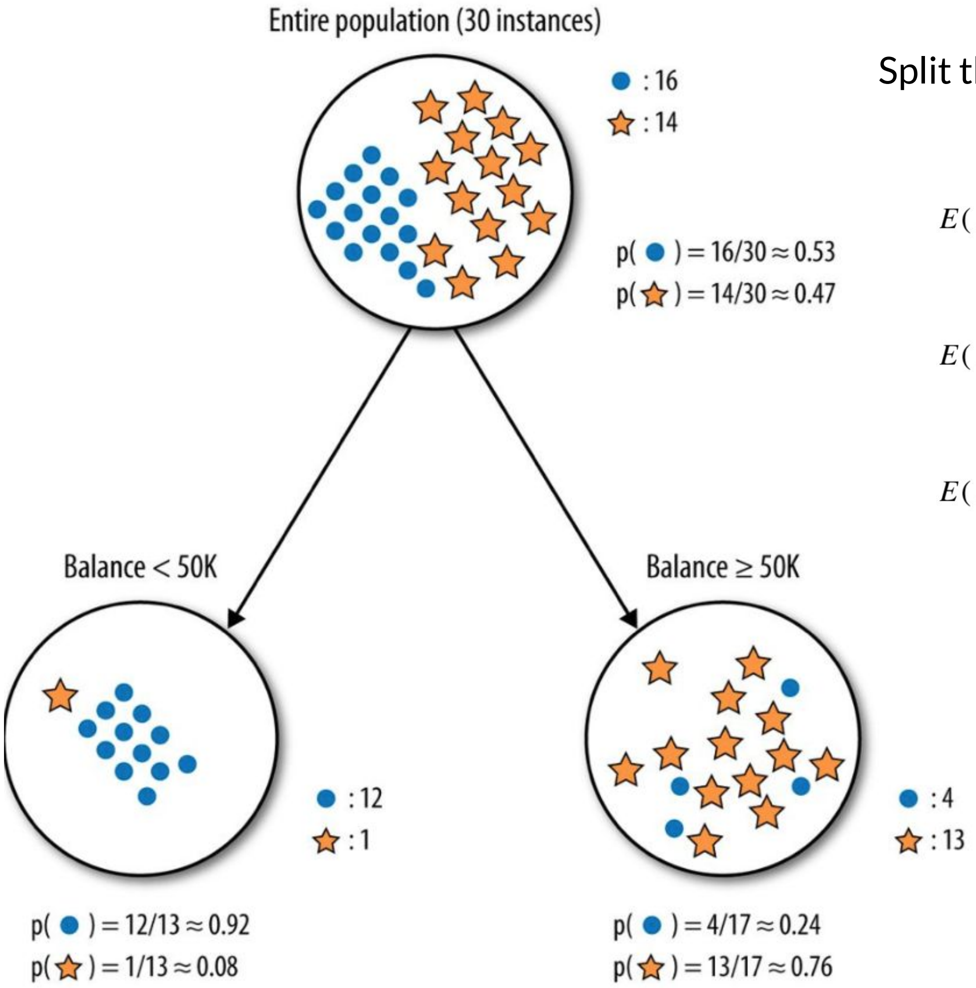
$$IG(T,X) = E(T) - E(T|X)$$

Information Gain from X on T

The information gain is based on the **decrease in entropy after a dataset is split** on an attribute.

Constructing a decision tree is all about finding attribute that returns the **highest information gain** (i.e., the most homogeneous branches).

Split the Decision Tree from feature "Balance"



$$E(Parent) = -\frac{16}{30}\log_2\left(\frac{16}{30}\right) - \frac{14}{30}\log_2\left(\frac{14}{30}\right) \approx 0.99$$

$$E(Balance < 50K) = -\frac{12}{13}\log_2\left(\frac{12}{13}\right) - \frac{1}{13}\log_2\left(\frac{1}{13}\right) \approx 0.39$$

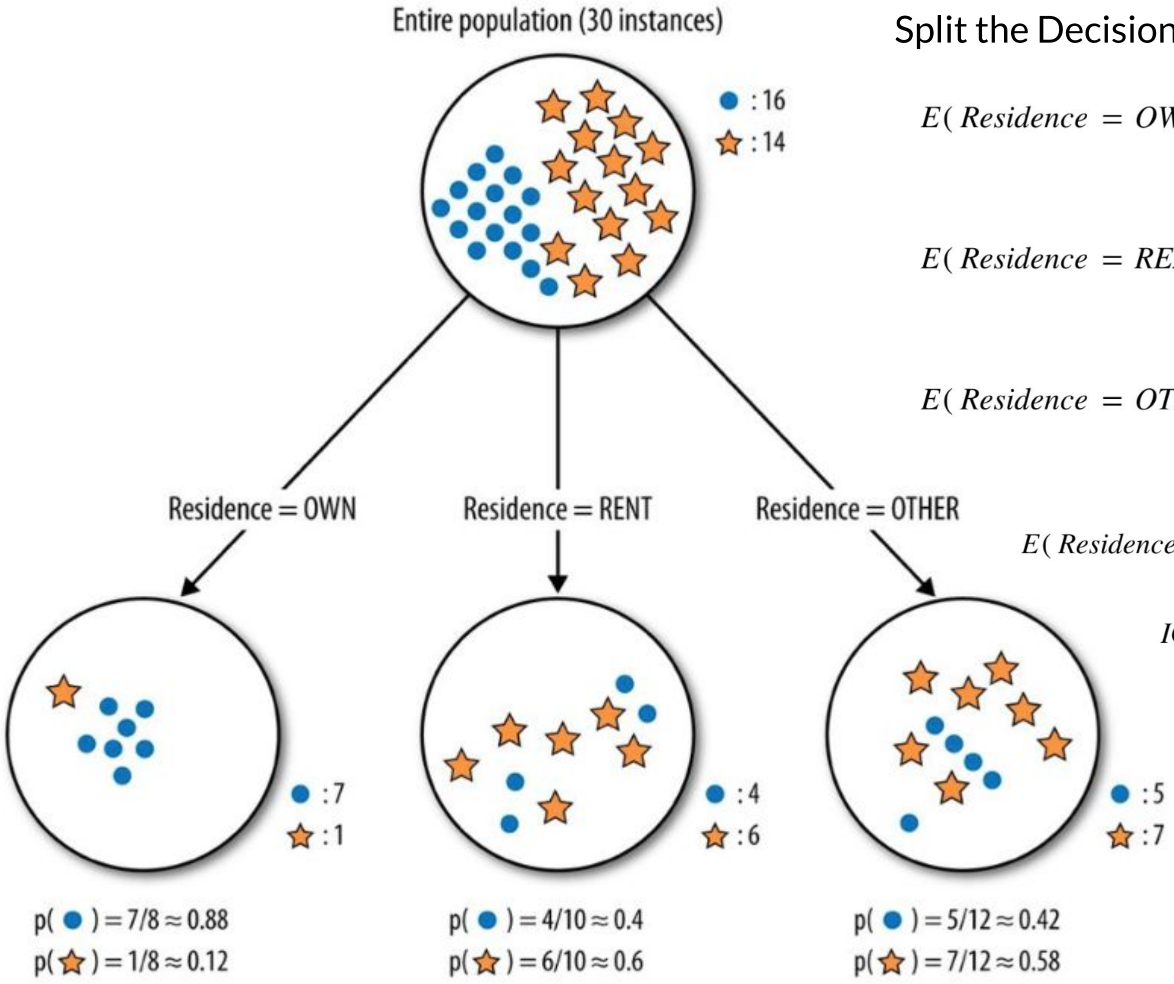
$$E(Balance > 50K) = -\frac{4}{17}\log_2\left(\frac{4}{17}\right) - \frac{13}{17}\log_2\left(\frac{13}{17}\right) \approx 0.79$$

$$\begin{aligned} E(Balance) &= \frac{13}{30}\times 0.39 + \frac{17}{30}\times 0.79 \\ &= 0.62 \end{aligned}$$

$$\begin{aligned} IG(Parent, Balance) &= E(Parent) - E(Balance) \\ &= 0.99 - 0.62 \\ &= 0.37 \end{aligned}$$

https://towardsdatascience.com/entropy-how-decision-trees-make-decisions-2946b9c18c8

Split the Decision Tree from feature "Residence"



$$E(\,Residence \;=\; OWN) \;=\; -\frac{7}{8}\log_2\left(\frac{7}{8}\right) - \frac{1}{8}\log_2\left(\frac{1}{8}\right) \approx 0.54$$

$$E(\,Residence \;=\; RENT\,) \;=\; -\frac{4}{10}\log_2\left(\frac{4}{10}\right) - \frac{6}{10}\log_2\left(\frac{6}{10}\right) \approx 0.97$$

$$E(\,Residence \;=\; OTHER) \;=\; -\frac{5}{12}\log_2\left(\frac{5}{12}\right) - \frac{7}{12}\log_2\left(\frac{7}{12}\right) \approx 0.98$$

$$E(\,Residence) \;=\frac{8}{30}\times 0.54 \;+\; \frac{10}{30}\times 0.97 \;+\; \frac{12}{30}\times 0.98 \;=\; 0.86$$

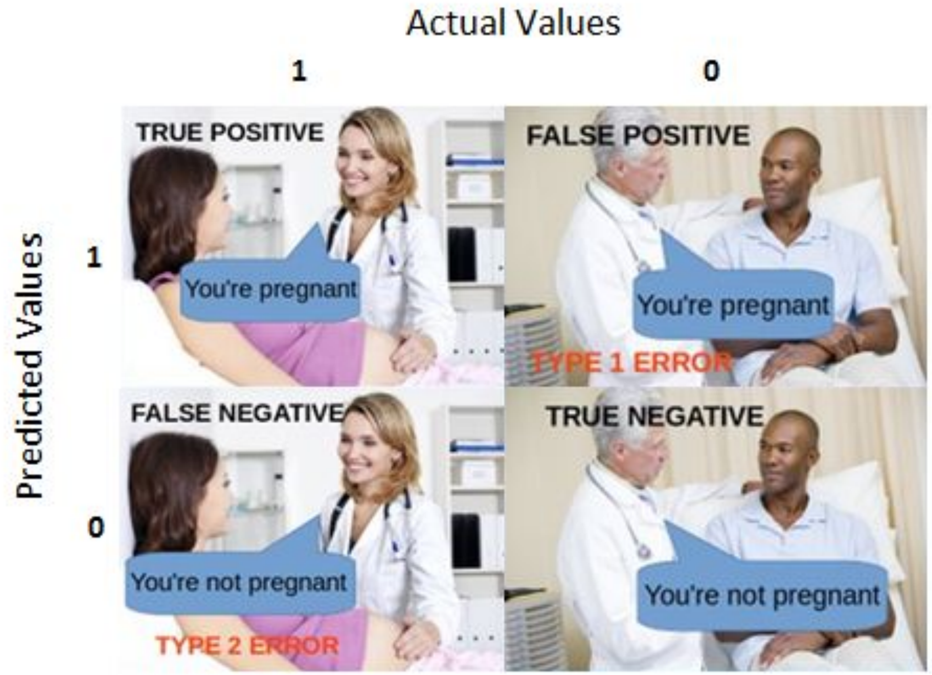
$$\begin{aligned} IG(\,Parent,\; Residence) &= E(\,Parent) \;-\; E(\,Residence) \\ &= 0.99 \;-\; 0.86 \\ &= 0.13 \end{aligned}$$

$$Gini(x) = 1 - \sum_{i=1}^{c} P(x_i)^2$$

$$Entropy(x) = -\sum_{i=1}^{c} P(x_i)\log_b P(x_i)$$

Gini index or Entropy is the criterion for calculating **Information Gain**. Both of them are measures of impurity of a node.

Evaluating Classifiers



Evaluating Binary Classifiers

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP}$$

$$TPR = \frac{\#\text{true positives}}{\#\text{true positives} + \#\text{false negatives}}$$

Recall

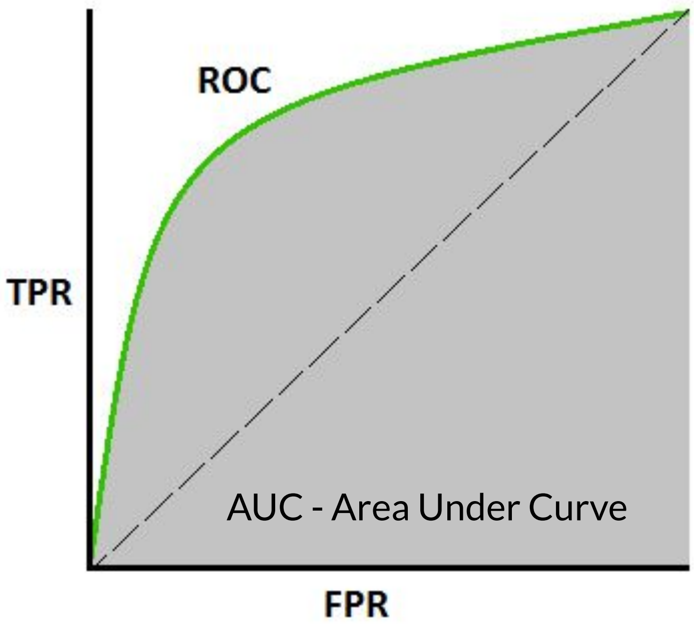
$$TNR = \frac{\#\text{true negatives}}{\#\text{true negatives} + \#\text{false positives}}$$

$$precision = \frac{TP}{(TP + FP)}$$

$$precision = \frac{TN}{(TN + FN)}$$

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

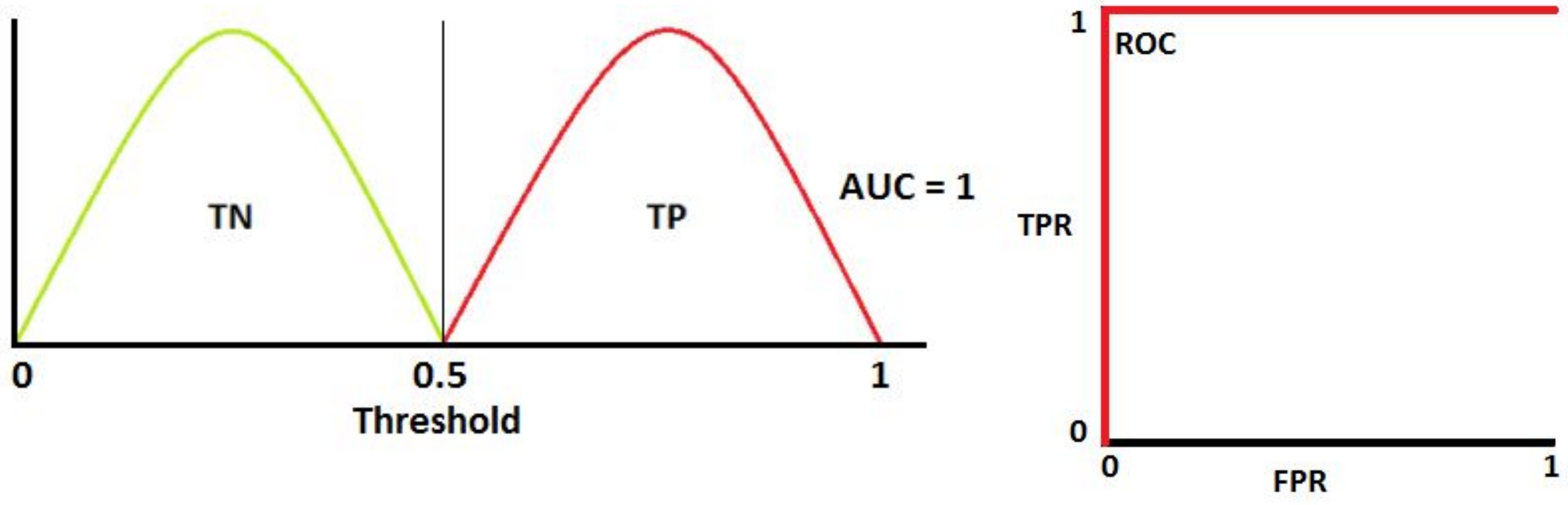
Receiver Operating Characteristic (ROC)



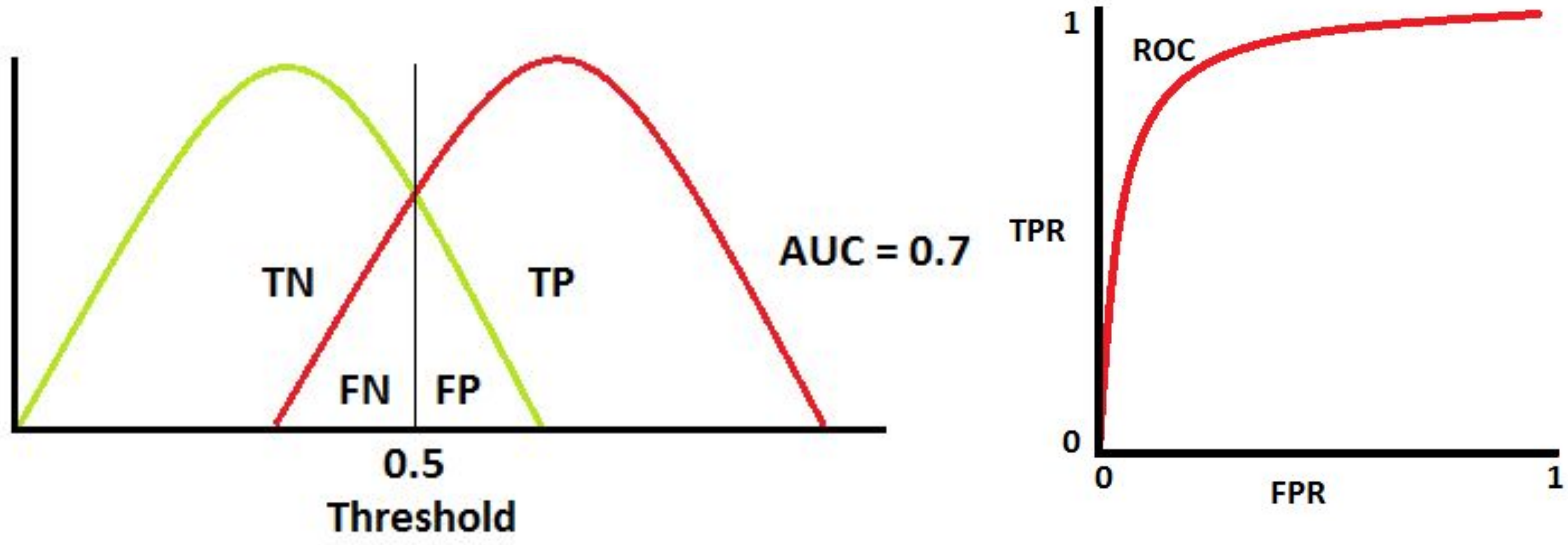
TPR
(measure the impact of True Positive)

FPR = 1 - TNR
(measure the impact of False Positive)

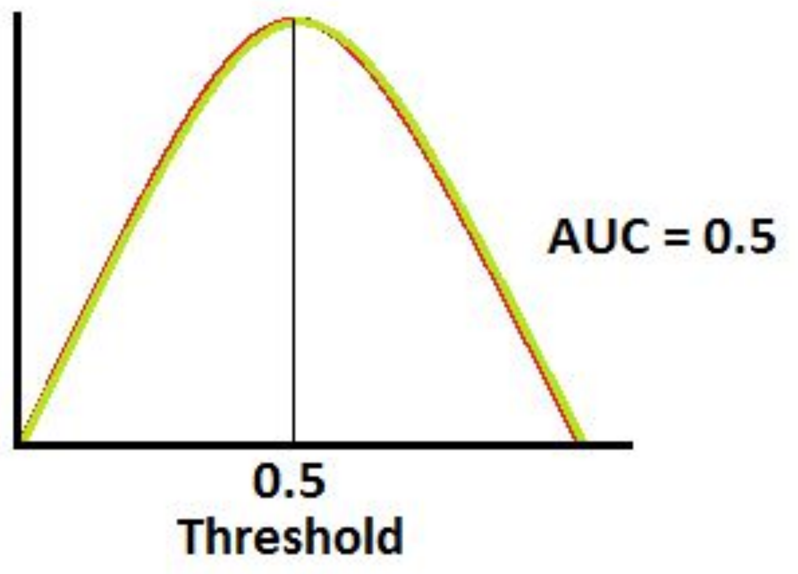
How to speculate the performance of model?

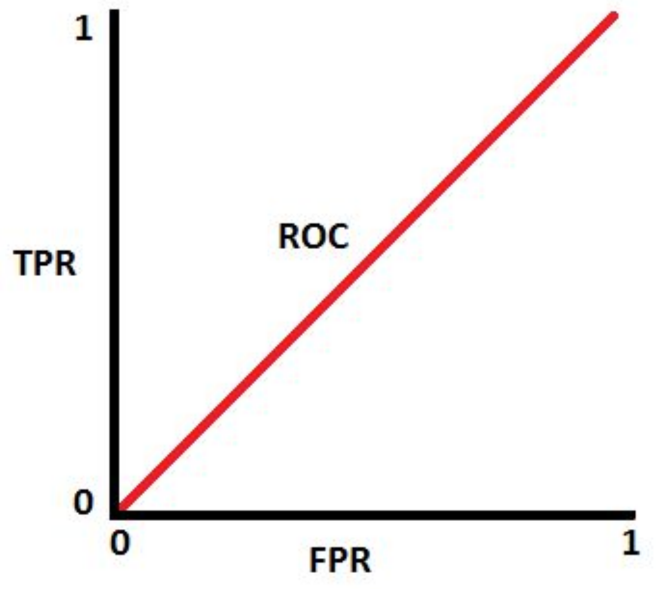


How to speculate the performance of model?

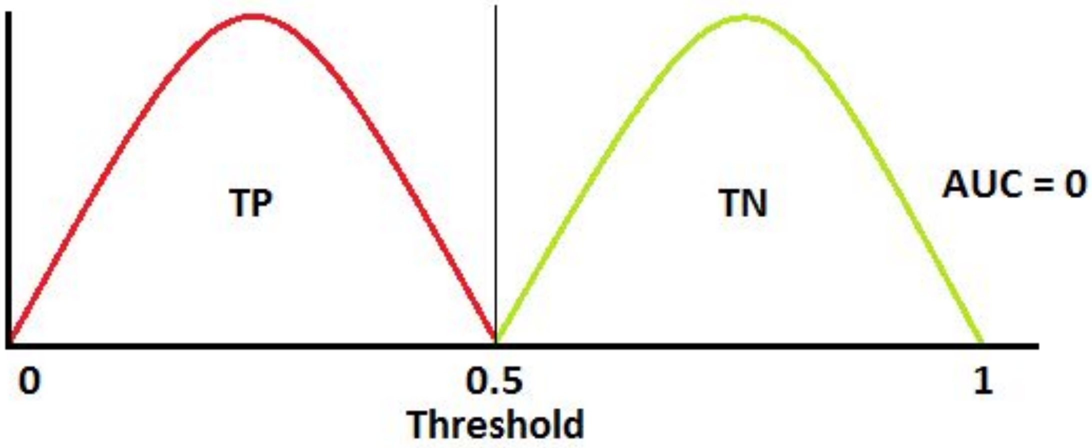


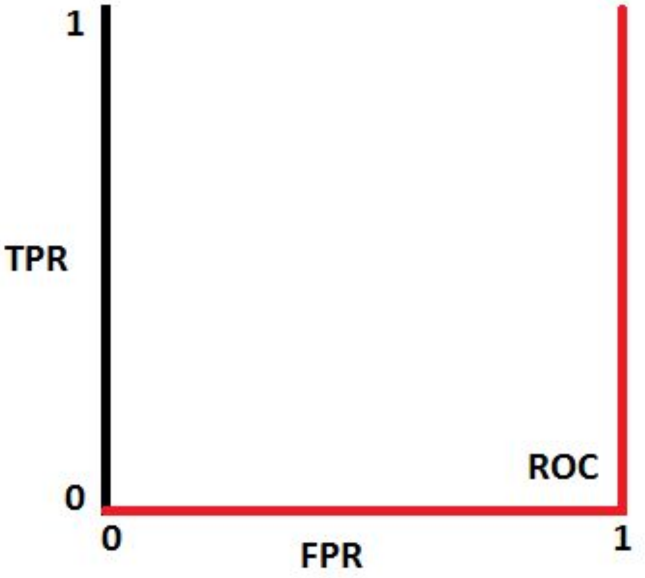
How to speculate the performance of model?





How to speculate the performance of model?





scikit learn

Applying Decision Tree

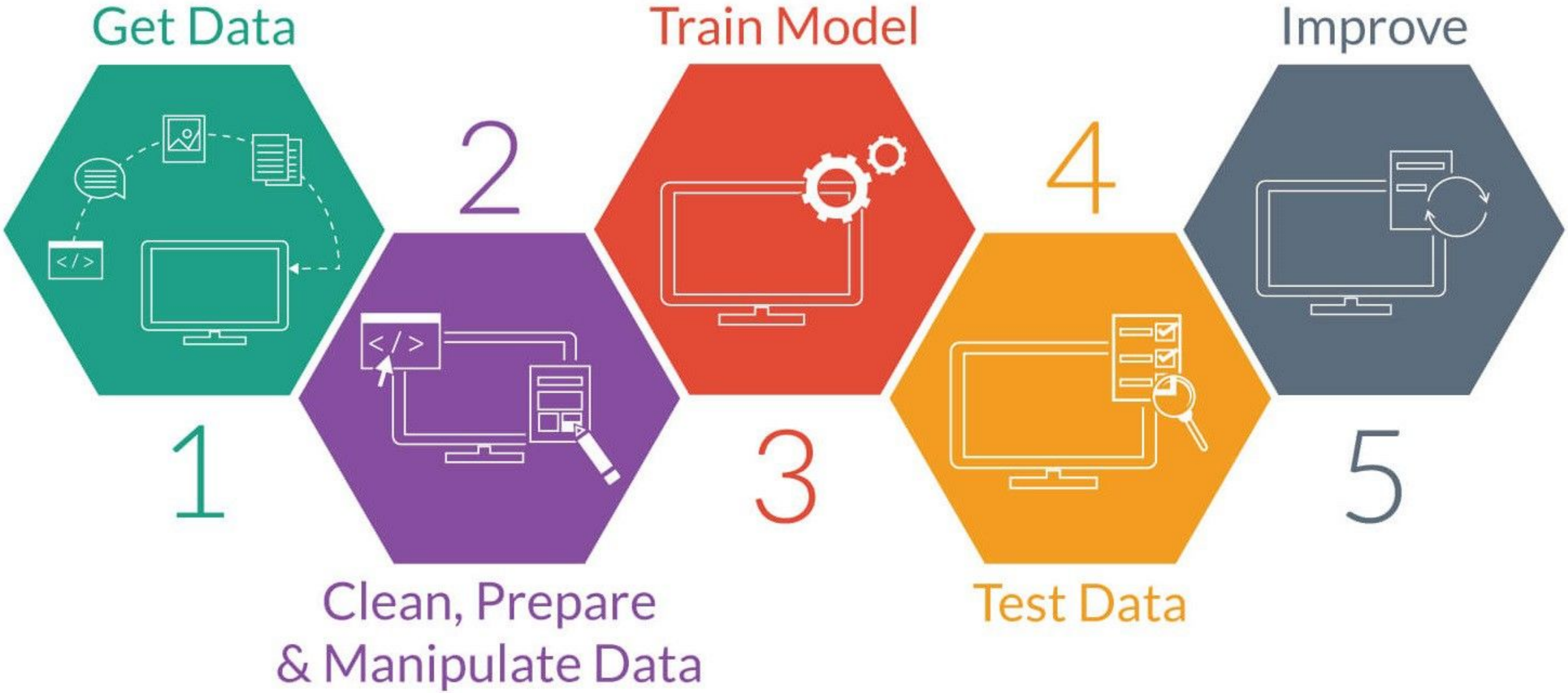
sklearn.tree.DecisionTreeClassifier

class `sklearn.tree.` **`DecisionTreeClassifier`** (*criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort=False*) [source]

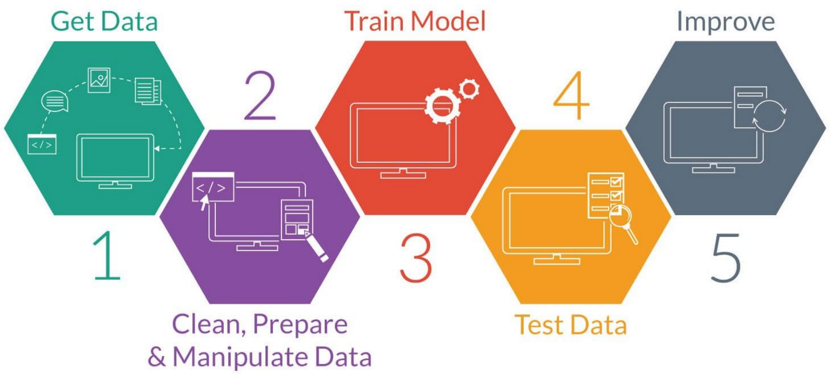
sklearn.tree.DecisionTreeRegressor

class `sklearn.tree.` **`DecisionTreeRegressor`** (*criterion='mse', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, presort=False*) ¶ [source]

A general ML workflow



A general ML workflow

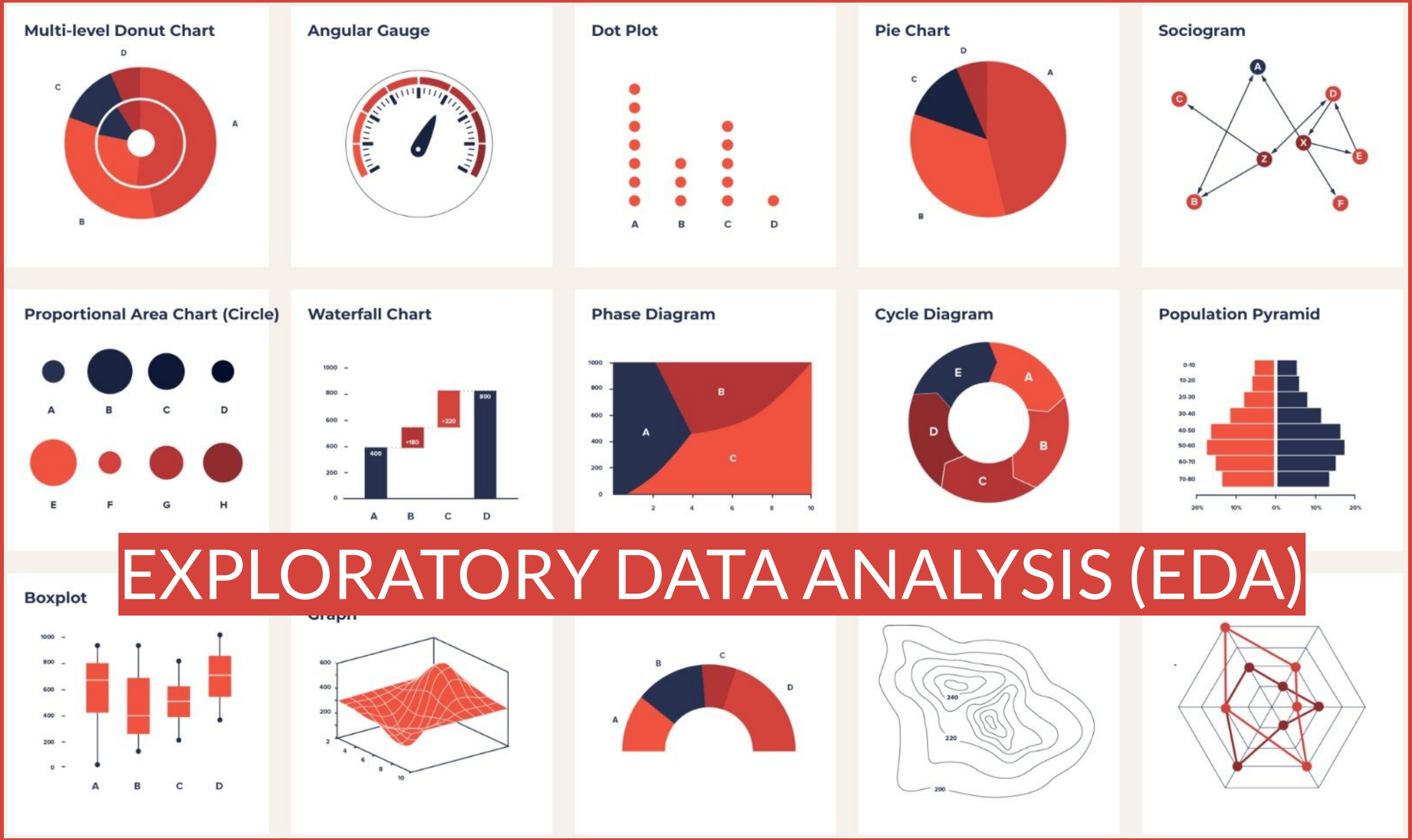


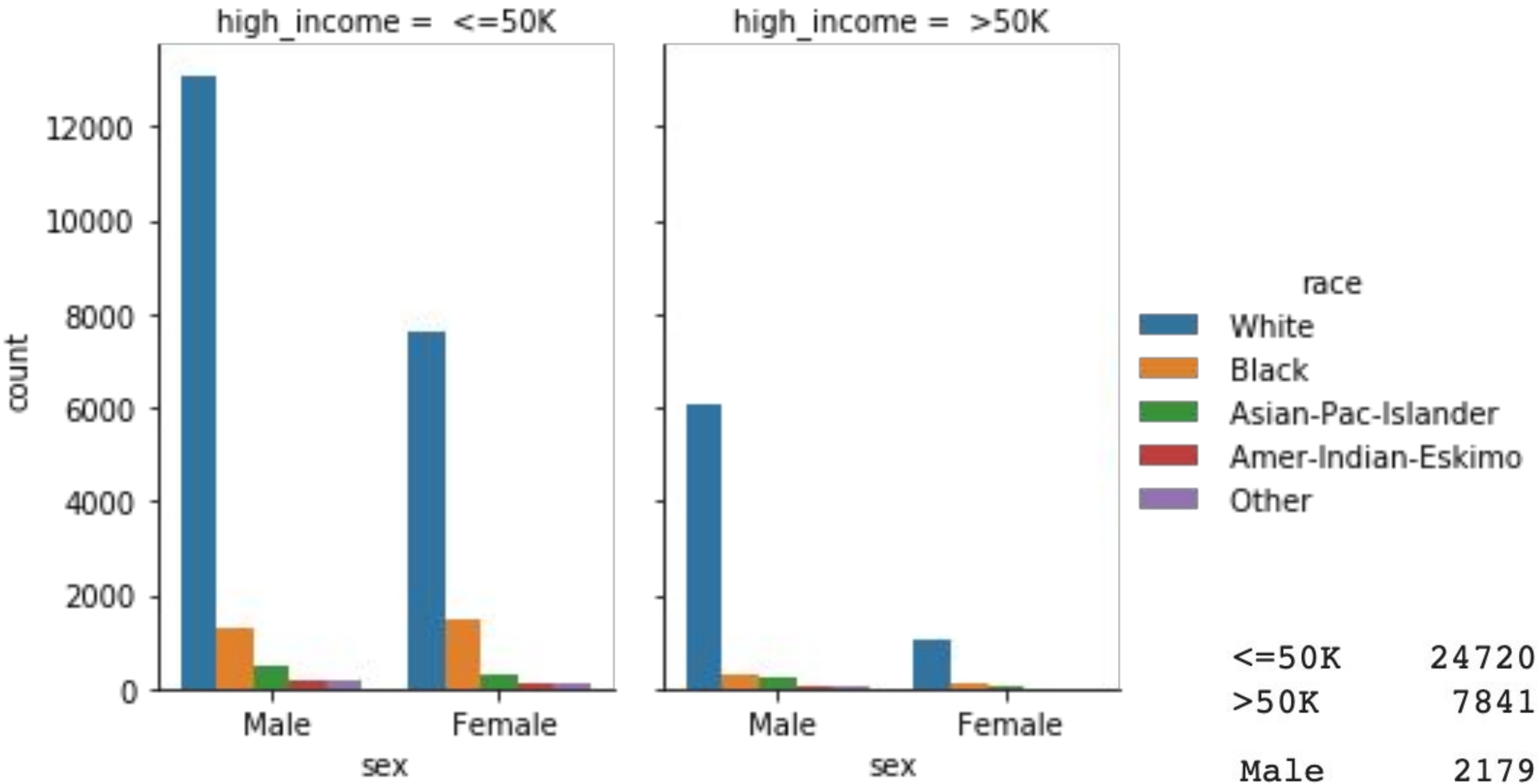
1. Load Libraries
2. Get data, including EDA
3. Clean, prepare and manipulate data (feature engineering)
4. Modeling (train and test)
5. Algorithm Tuning
6. Finalizing the Model

Load Libraries

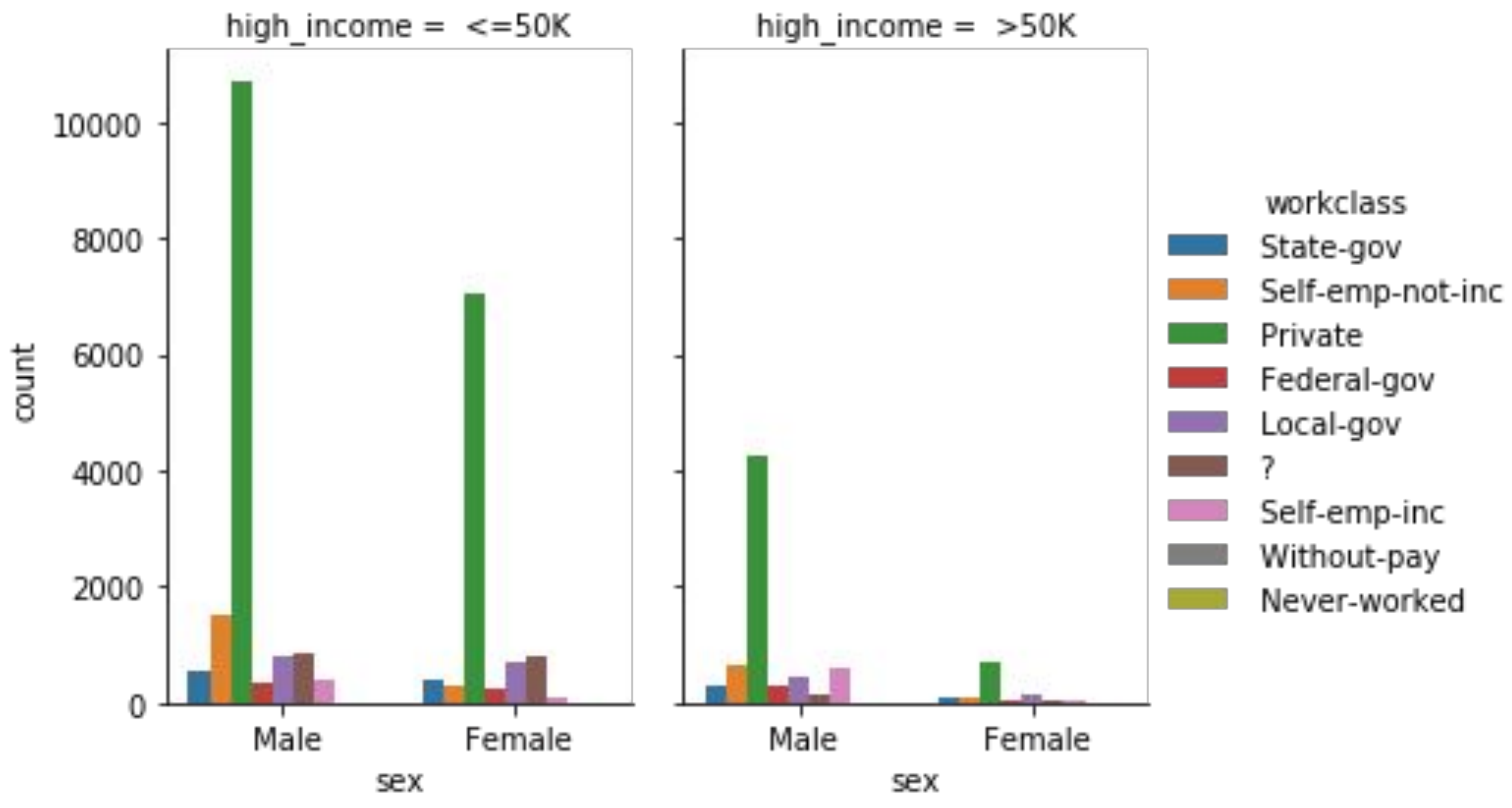
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
import pydotplus
from IPython.display import Image
from sklearn import tree
```

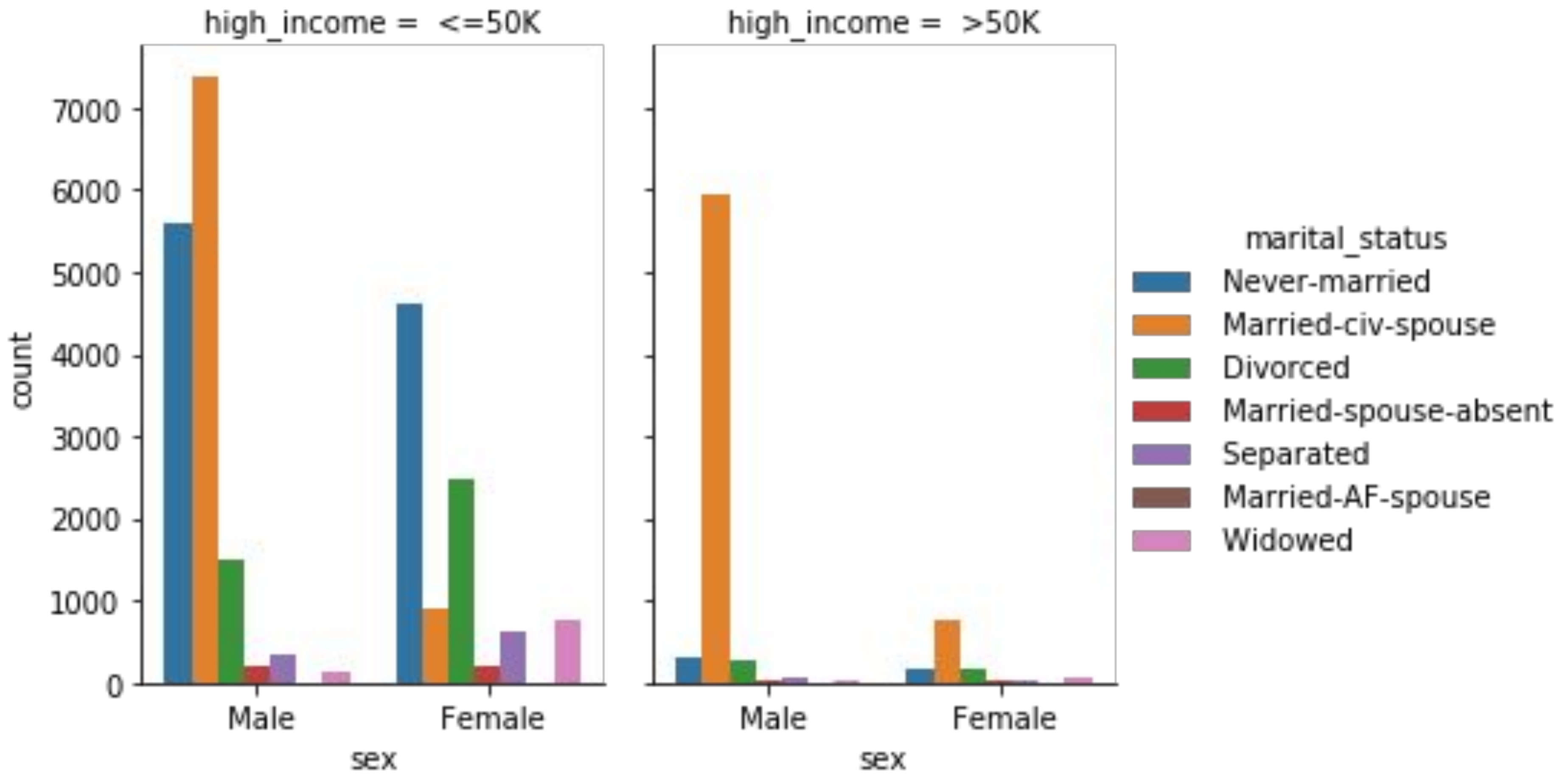
EXPLORATORY DATA ANALYSIS (EDA)

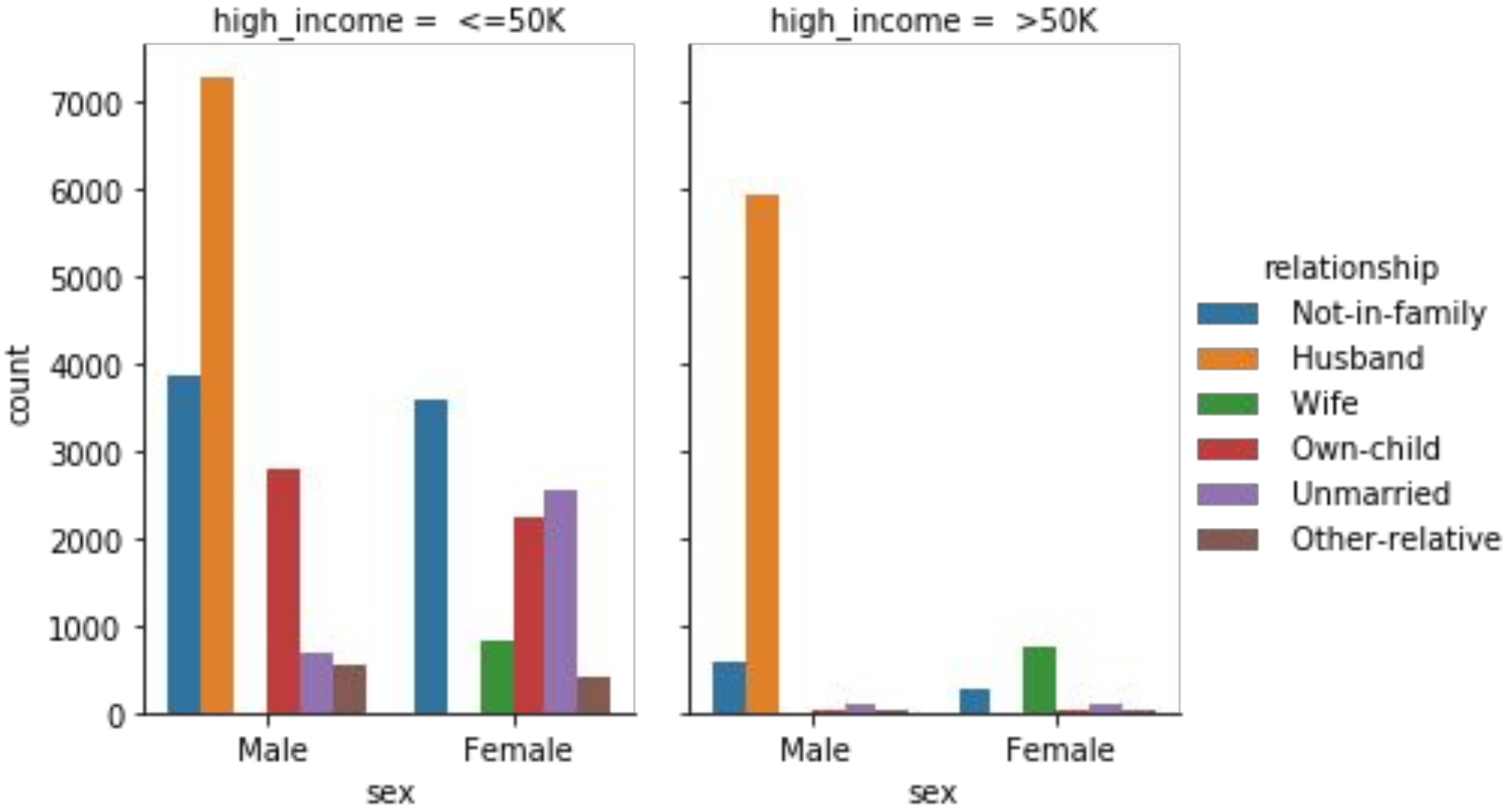


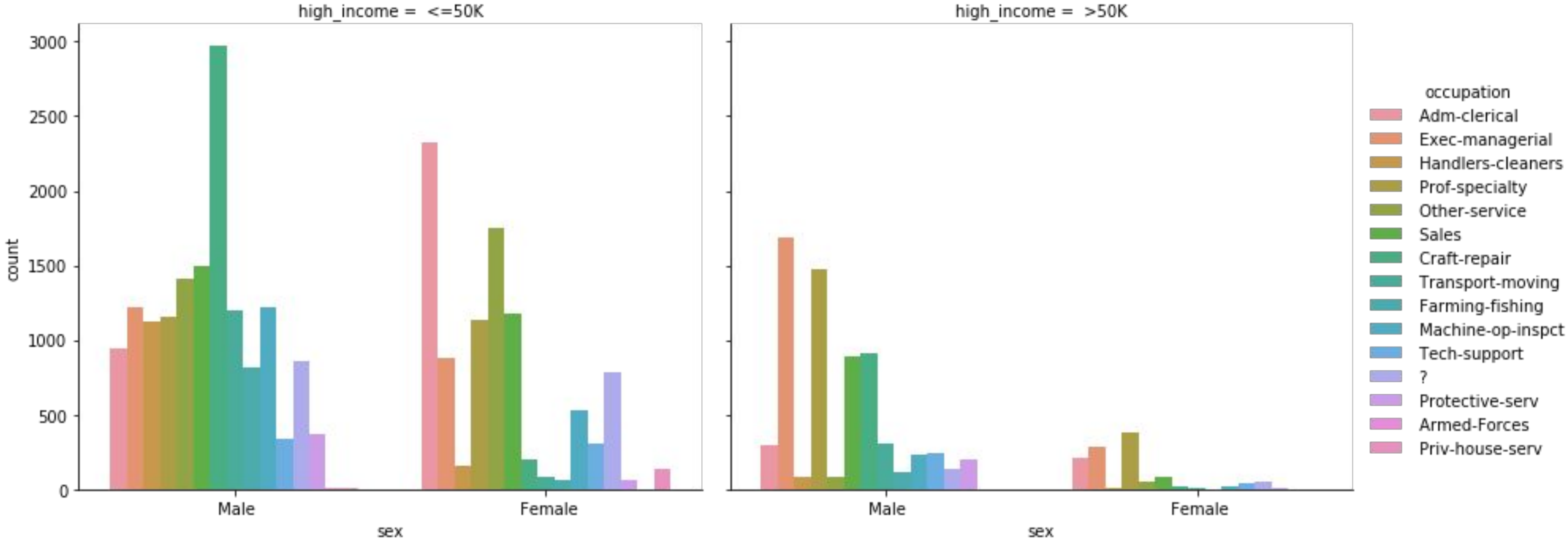


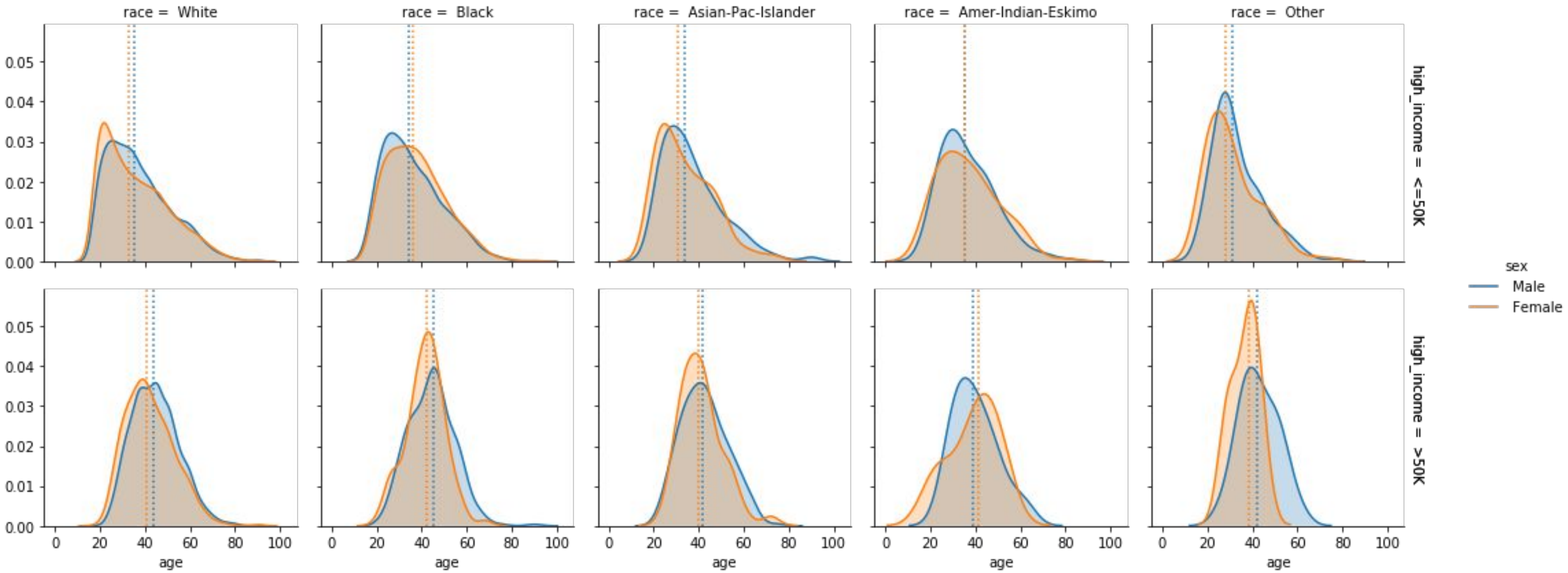

```
<=50K    24720
>50K      7841

Male      21790
Female    10771
```


high_income = <=50K
high_income = >50K
count
10000
8000
6000
4000
2000
0
Male
Female
sex
workclass
State-gov
Self-emp-not-inc
Private
Federal-gov
Local-gov
?
Self-emp-inc
Without-pay
Never-worked

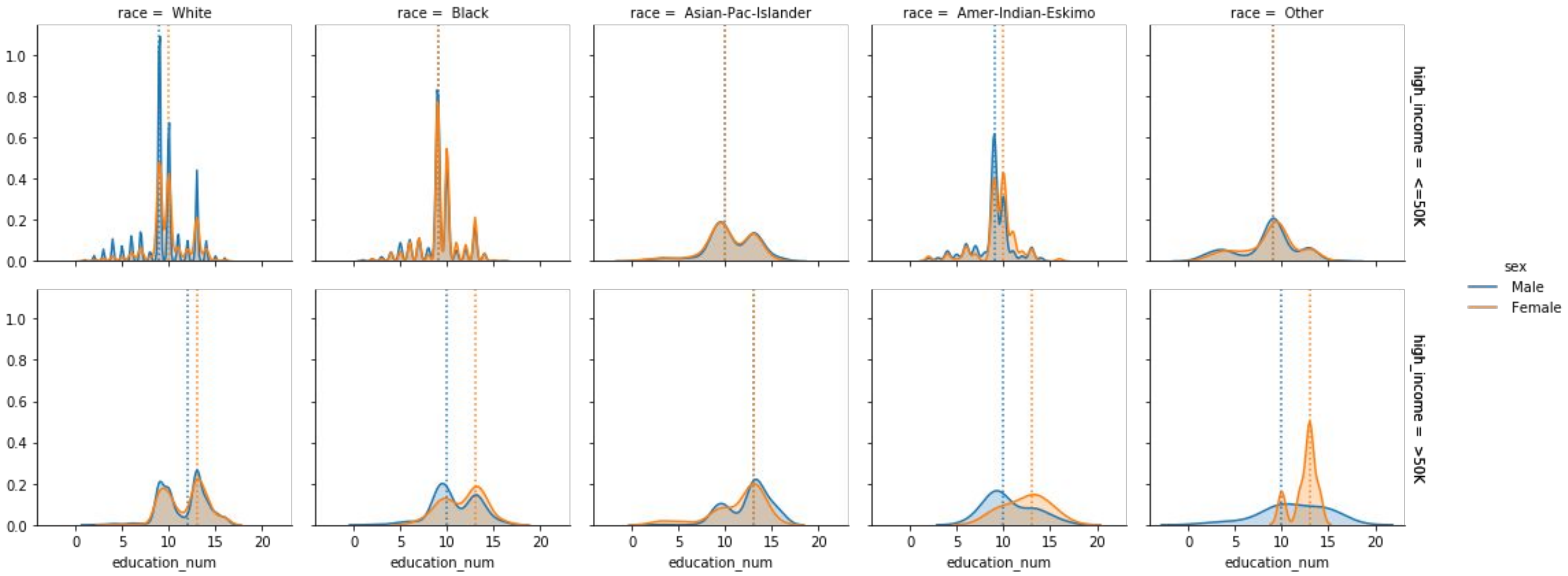

high_income = <=50K
high_income = >50K
count
7000
6000
5000
4000
3000
2000
1000
0
Male
Female
sex
marital_status
Never-married
Married-civ-spouse
Divorced
Married-spouse-absent
Separated
Married-AF-spouse
Widowed


high_income = <=50K
high_income = >50K
count
7000
6000
5000
4000
3000
2000
1000
0
Male
Female
sex
relationship
Not-in-family
Husband
Wife
Own-child
Unmarried
Other-relative


high_income = <=50K
high_income = >50K
count
3000
2500
2000
1500
1000
500
0
Male
Female
sex
occupation
Adm-clerical
Exec-managerial
Handlers-cleaners
Prof-specialty
Other-service
Sales
Craft-repair
Transport-moving
Farming-fishing
Machine-op-inspct
Tech-support
?
Protective-serv
Armed-Forces
Priv-house-serv



race = White
race = Black
race = Asian-Pac-Islander
race = Amer-Indian-Eskimo
race = Other
high_income = <=50K
high_income = >50K
sex
Male
Female
age
0.00
0.01
0.02
0.03
0.04
0.05
0
20
40
60
80
100

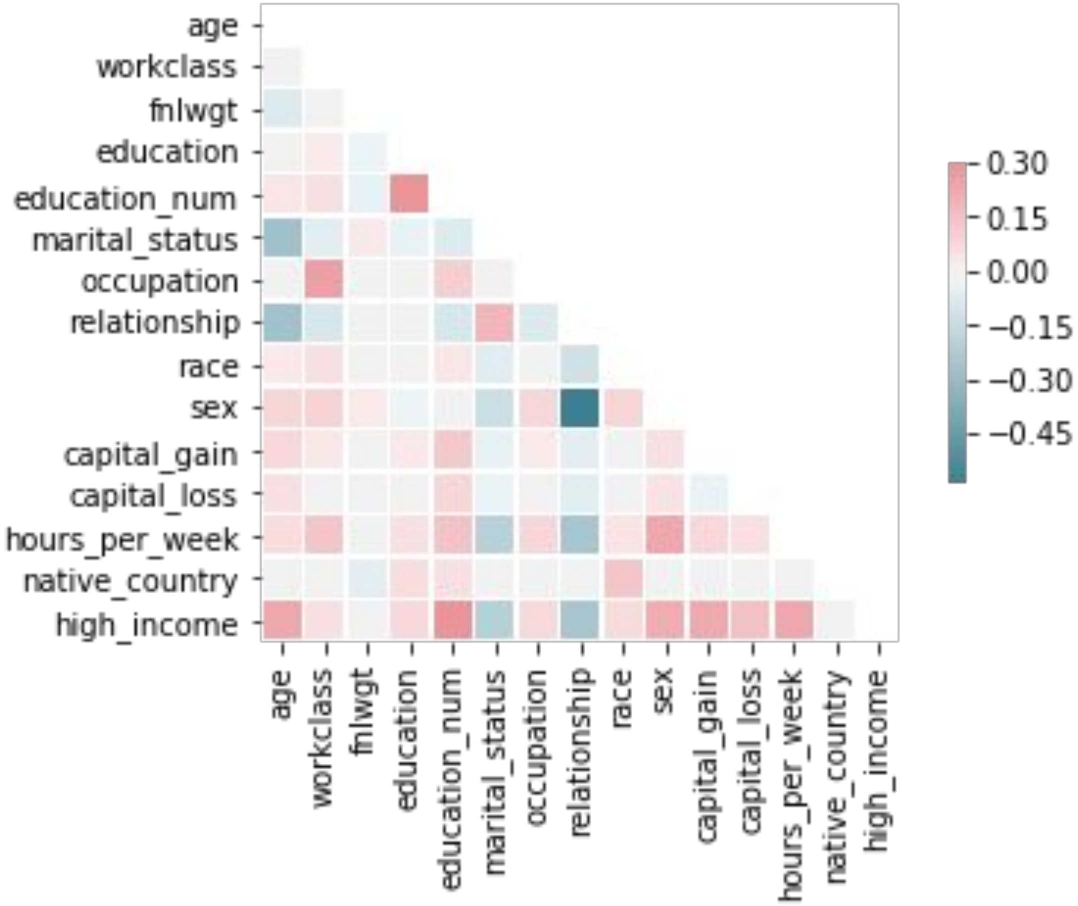

race = White
race = Black
race = Asian-Pac-Islander
race = Amer-Indian-Eskimo
race = Other
high_income = <=50K
high_income = >50K
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
education_num
sex
Male
Female

CLEAN, PREPARE AND MANIPULATE DATA

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	high_income
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	<=50K

```python
for name in income.select_dtypes("object").columns.to_list():
    col = pd.Categorical(income[name])
    income[name] = col.codes
```

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	high_income
0	39	7	77516	9	13	4	1	1	4	1	0
1	50	6	83311	9	13	2	4	0	4	1	0
2	38	4	215646	11	9	0	6	1	4	1	0
3	53	4	234721	1	7	2	6	0	2	1	0
4	28	4	338409	9	13	2	10	5	2	0	0



high_income	1.000000
education_num	0.335154
age	0.234037
hours_per_week	0.229689
capital_gain	0.223329
sex	0.215980
capital_loss	0.150526
education	0.079317
occupation	0.075468
race	0.071846
workclass	0.051604
native_country	0.015840
fnlwgt	-0.009463
marital_status	-0.199307
relationship	-0.250918

Train and Test

```python
# split-out train/validation and test dataset
X_train, X_test, y_train, y_test = train_test_split(income.drop(labels="high_income",axis=1),
                                                    income["high_income"],
                                                    test_size=0.20,
                                                    random_state=seed,
                                                    shuffle=True,
                                                    stratify=income["high_income"])
```

Algorithm Tuning

```
# create a pipeline
pipe = Pipeline([("classifier",DecisionTreeClassifier())])

# create a dictionary with the hyperparameters
search_space = [{"classifier":[DecisionTreeClassifier()],
                 "classifier__criterion": ["gini","entropy"]},
                {"classifier":[LogisticRegression()],
                 "classifier__solver": ["liblinear"]},
                {"classifier": [KNeighborsClassifier()],
                 "classifier__n_neighbors": [5,7]}]

# create grid search
kfold = KFold(n_splits=num_folds,random_state=seed)
grid = GridSearchCV(estimator=pipe,
                    param_grid=search_space,
                    cv=kfold,
                    scoring=scoring,
                    n_jobs=-1)

# fit grid search
best_model = grid.fit(X_train,y_train)
```

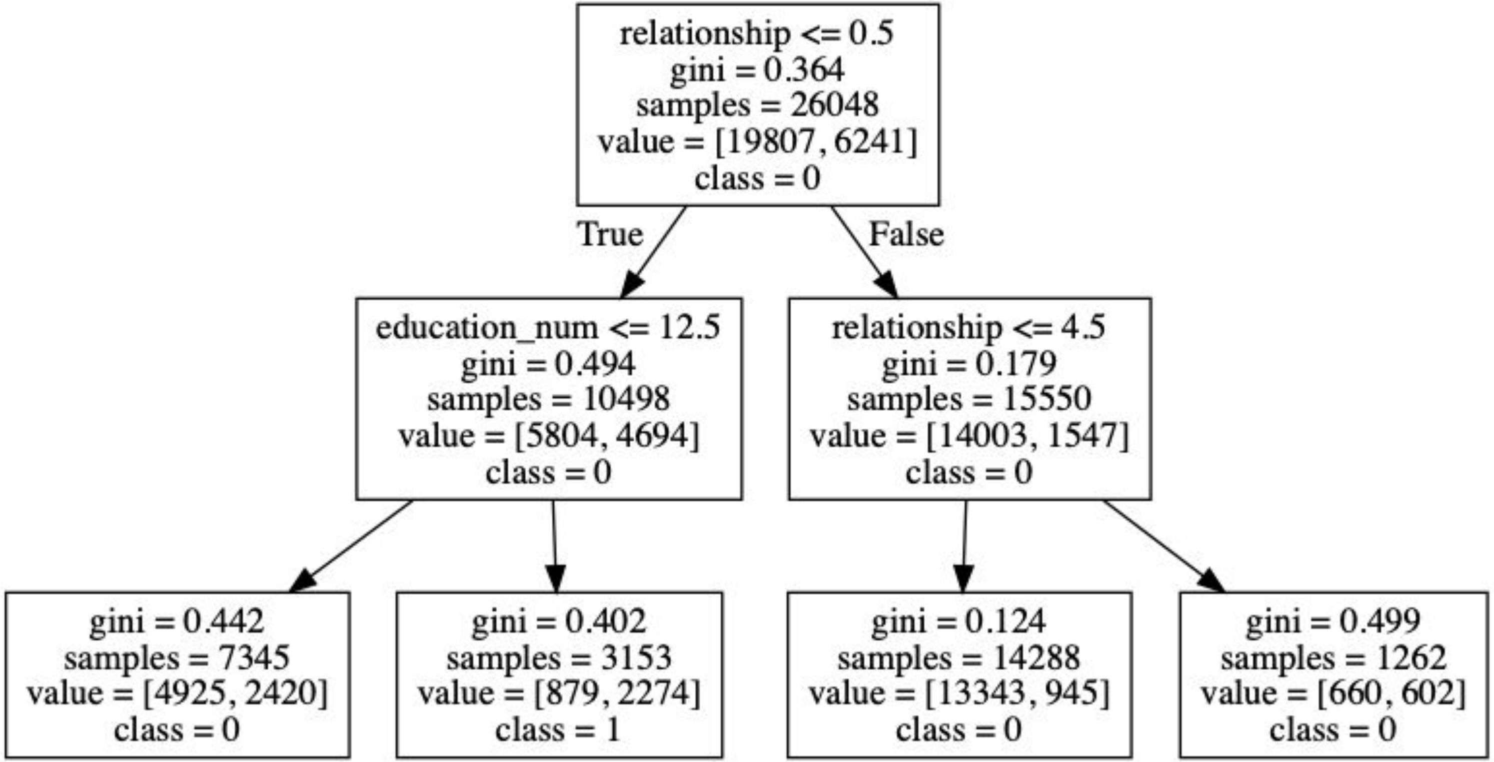
```
0.807049 (0.004056) with: {'classifier': DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best'), 'classifier__criterion': 'gini'}
0.814727 (0.008882) with: {'classifier': DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best'), 'classifier__criterion': 'entropy'}
0.794725 (0.012523) with: {'classifier': LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False), 'classifier__solver': 'liblinear'}
0.774109 (0.008792) with: {'classifier': KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform'), 'classifier__n_neighbors': 5}
0.783745 (0.009897) with: {'classifier': KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform'), 'classifier__n_neighbors': 7}
```

Finalize the Model

```python
# final model
predict = best_model.predict(X_test)
print(accuracy_score(y_test, predict))
print(confusion_matrix(y_test,predict))
print(classification_report(y_test,predict))
```

```
0.8191309688315677
[[4342  603]
 [ 575  993]]
              precision    recall  f1-score   support

           0       0.88      0.88      0.88      4945
           1       0.62      0.63      0.63      1568

    accuracy                           0.82      6513
   macro avg       0.75      0.76      0.75      6513
weighted avg       0.82      0.82      0.82      6513
```

Visualizing a Decision Tree Model



Knowing when to use decision trees

The main advantages of using decision trees is that they're:

- Easy to interpret
- Relatively fast to fit and make predictions
- Able to handle multiple types of data
- Able to pick up nonlinearities in data, and usually fairly accurate

The main disadvantage of using decision trees is their **tendency to overfit.**

Lesson #17 - Decision Trees.ipynb