Machine Learning - Trees

Created and Edited by Amanda Magzal.

Table of Contents

---

1 Libraries

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# decision tree
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor

# random forest
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor

# model building
from sklearn.model_selection import train_test_split, GridSearchCV

# model evaluation
from sklearn.metrics import accuracy_score, mean_squared_error

# set seed for reproducible results
RSEED = 10
```

---

2 Data

The dataset consists of thre Titanic's passengers information. It can be found here.

Variables:

- **PassengerId** - the id of the passenger (unique number).
- **Survived** - if the person survived (0 = No, 1 = Yes).
- **Pclass** - ticket class (1 = 1st class, 2 = 2nd class, 3 = 3rd class).
- **Name** - passenger name.
- **Sex** - the sex of the passenger (male, female).
- **SibSp** - the number of siblings or spouses aboard the Titanic.
- **Parch** - the number of parents or children abroad the Titanic.
- **Ticket** - ticket number.
- **Fare** - passenger fare.
- **Cabin** - passenger cabin number.
- **Embarked** - port of embarkation (C = Cherbourg, Q = Queenstown, S = Southampton).

In [2]:

```
# import the data
df = pd.read_csv('titanic-data.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [4]:

```
df.shape
```

Out[4]:

```
(891, 12)
```

In [5]:

```
# drop irrelevant features
df.drop(['PassengerId', 'Name', 'SibSp', 'Parch', 'Ticket', 'Cabin', 'Embarked'], axis=1, inplace=True)
```

In [6]:

```
# identify missing values
df.isnull().sum()
```

Out[6]:

```
Survived      0
Pclass        0
Sex           0
Age         177
Fare          0
dtype: int64
```

In [7]:

```
# fill in missing values
df['Age'] = df['Age'].fillna(df['Age'].mean())
```

In [8]:

```
# one-hot encoding
df = pd.get_dummies(df, columns=['Sex'], drop_first=True)
```

Classification Data

For classification algorithms, we are interested in predicting whether a passenger survived - the variable `Survived`.

In [9]:

```
X_cls = df[['Pclass', 'Sex_male', 'Age', 'Fare']]
y_cls = df['Survived']
```

In [10]:

```
# Split dataset into training set and test set
X_cls_train, X_cls_test, y_cls_train, y_cls_test = train_test_split(X_cls, y_cls, test_size=0.3, random_state=RSE
```

Regression Data

For regression algorithms, we are interested in predicting a passenger's age - the variable `Age`.

In [11]:

```
X_reg = df[['Survived', 'Pclass', 'Sex_male', 'Fare']]
y_reg = df['Age']
```

In [12]:

```
# Split dataset into training set and test set
X_reg_train, X_reg_test, y_reg_train, y_reg_test = train_test_split(X_reg, y_reg, test_size=0.3, random_state=RSE
```

---

3 Decision Tree

3.1 Classification

Define the Model

In [13]:

```
# define model
dt = DecisionTreeClassifier(random_state=RSEED)

# define parameter grid
parameters_grid = {
    'max_depth': [2, 3],
    'min_samples_leaf': [2, 8],
    'max_features': [2, 4]
}

# define grid search
grid_search = GridSearchCV(estimator=dt, param_grid=parameters_grid, cv=10)
```

Fit the Model

In [14]:

```
# fit estimator
grid_search.fit(X_cls_train, y_cls_train)

# get best estimator
best = grid_search.best_estimator_
```

In [15]:

```
# print best parameters
pd.DataFrame.from_dict(grid_search.best_params_, orient='index', columns=['Selected Value']).T
```

Out[15]:

	max_depth	max_features	min_samples_leaf
Selected Value	3	4	2

Predict

In [16]:

```
# predict
y_pred = best.predict(X_cls_test)
```

Evaluate the Model

In [17]:

```
# calculate accuracy
acc = round(accuracy_score(y_cls_test, y_pred), 3)

df = pd.DataFrame([acc]).T
df = df.rename(index={0: 'Decision Tree Classifier'}, columns={0: 'Accuracy'})
df
```

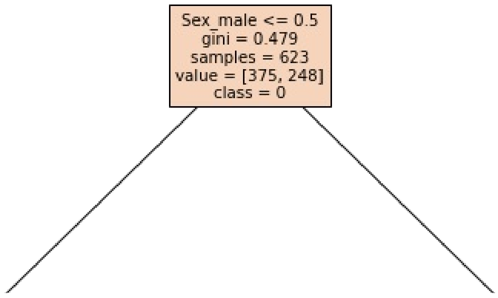
Out[17]:

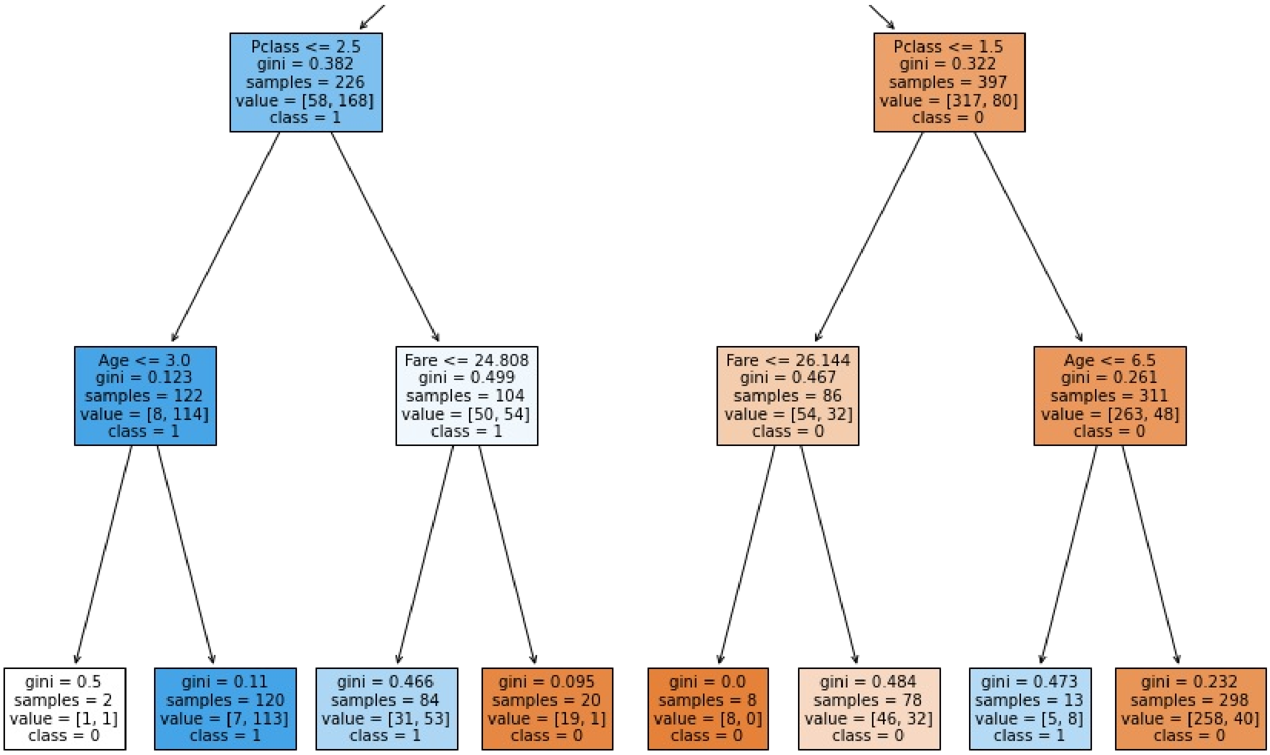
	Accuracy
Decision Tree Classifier	0.817

Visualizing the Tree

In [18]:

```
plt.figure(figsize=(15, 15))
tr = tree.plot_tree(best, feature_names=X_cls_train.columns, class_names=['0', '1'], filled=True)
```





3.2 Regression

Define the Model

In [19]:

```
# define model
dt = DecisionTreeRegressor(random_state=RSEED)

# define parameter grid
parameters_grid = {
    'max_depth': [2, 3],
    'min_samples_leaf': [2, 8],
    'max_features': [2, 4]
}

# define grid search
grid_search = GridSearchCV(estimator=dt, param_grid=parameters_grid, cv=10)
```

Fit the Model

In [20]:

```
# fit estimator
grid_search.fit(X_reg_train, y_reg_train)

# get best estimator
best = grid_search.best_estimator_
```

In [21]:

```
# print best parameters
pd.DataFrame.from_dict(grid_search.best_params_, orient='index', columns=['Selected Value']).T
```

Out[21]:

	max_depth	max_features	min_samples_leaf
Selected Value	3	4	2

Predict

In [22]:

```
# predict
y_pred = best.predict(X_reg_test)
```

Evaluate the Model

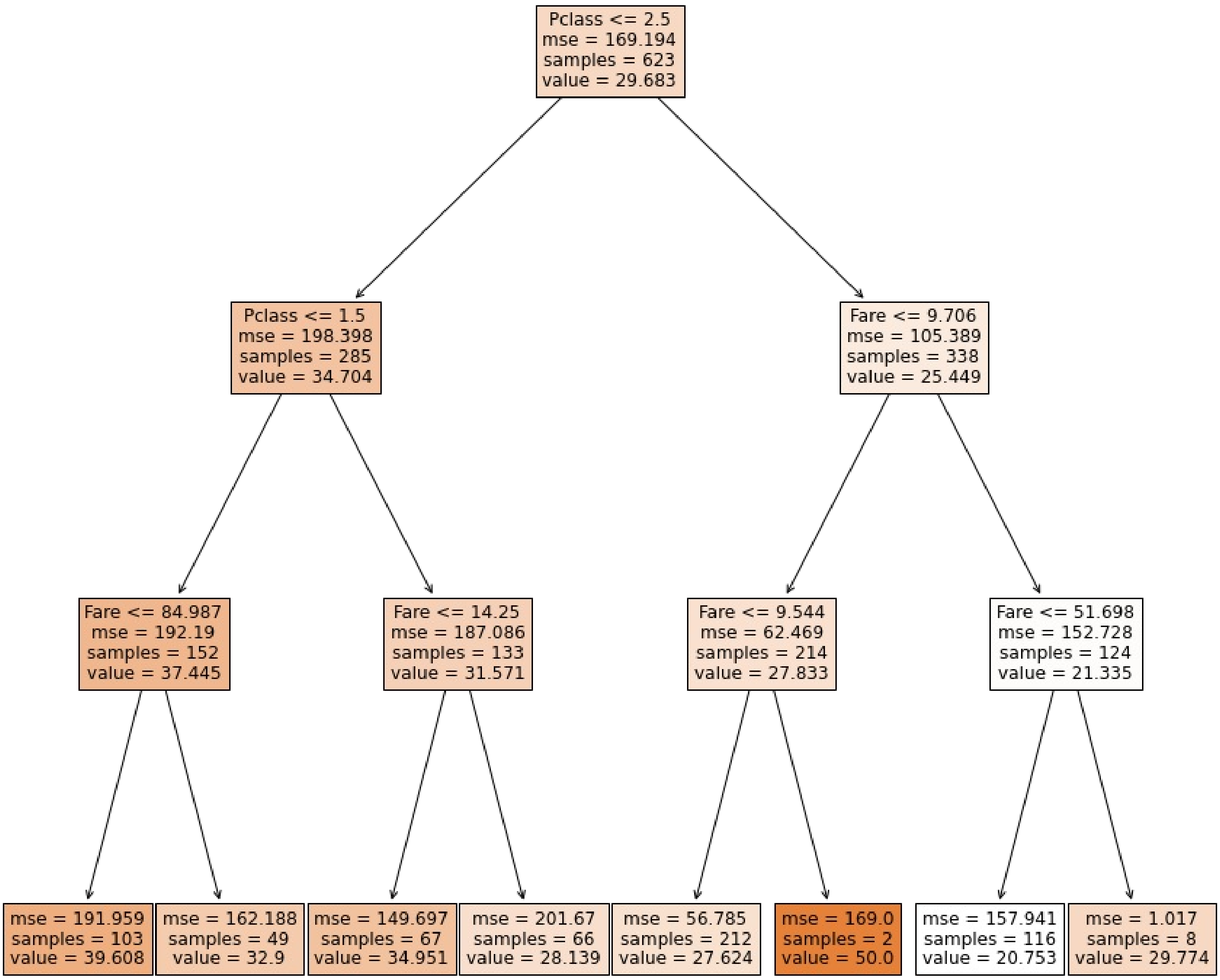
In [23]:
```
# calculate MSE
MSE = round(mean_squared_error(y_reg_test, y_pred), 3)

df = pd.DataFrame([MSE]).T
df = df.rename(index={0: 'Decision Tree Regressor'}, columns={0: 'MSE'})
df
```

Out[23]:

	MSE
Decision Tree Regressor	158.738

Visualizing the Tree

In [24]:
```
plt.figure(figsize=(15, 15))
tr = tree.plot_tree(best, feature_names=X_reg_train.columns, filled=True)
```



---

4 Random Forest

4.1 Classification

Define the Model

In [25]:
```
# define model
rf = RandomForestClassifier(random_state=RSEED)

# define parameter grid
parameters_grid = {
    'max_depth': [3, 5],
    'min_samples_leaf': [2, 8],
    'n_estimators': [20, 50],
    'max_features': [2, 4]
}

# define grid search
grid_search = GridSearchCV(estimator=rf, param_grid=parameters_grid, cv=10)
```

Fit the Model

In [26]:
```
# fit estimator
grid_search.fit(X_cls_train, y_cls_train)

# get best estimator
best = grid_search.best_estimator_
```

In [27]:
```
# print best parameters
pd.DataFrame.from_dict(grid_search.best_params_, orient='index', columns=['Selected Value']).T
```

Out[27]:

	max_depth	max_features	min_samples_leaf	n_estimators
Selected Value	5	4	2	20

Predict

In [28]:
```
# predict
y_pred = best.predict(X_cls_test)
```

Evaluate the Model

In [29]:
```
# calculate accuracy
acc = round(accuracy_score(y_cls_test, y_pred), 3)

df = pd.DataFrame([acc]).T
df = df.rename(index={0: 'Random Forest Classifier'}, columns={0: 'Accuracy'})
df
```

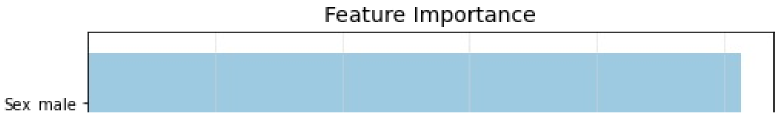
Out[29]:

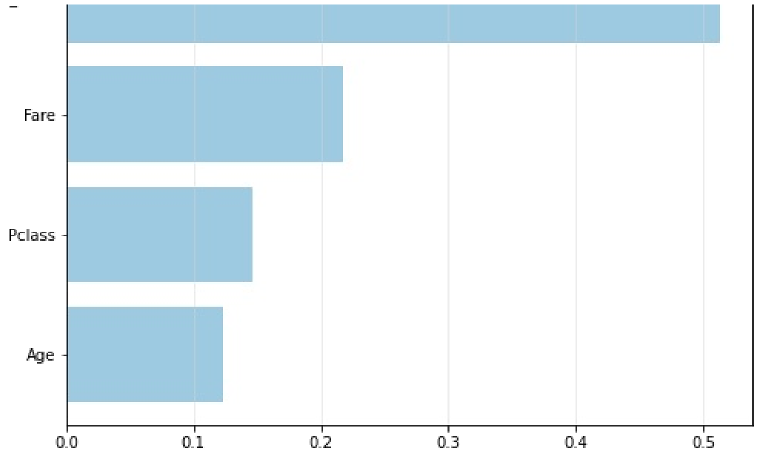
	Accuracy
Random Forest Classifier	0.813

Feature Importance

In [30]:
```
# get feature importance
imp = pd.DataFrame(grid_search.best_estimator_.fit(X_cls_train, y_cls_train).feature_importances_,
                   index=X_cls_train.columns, columns=['Importance']).sort_values('Importance')
```

In [31]:
```
# plot feature importance
plt.figure(figsize=(8, 6))
plt.barh(range(0, len(imp)), imp['Importance'], color='#9ecae1')
plt.grid(axis='x', alpha=0.5, color='lightgrey')
plt.yticks(range(0, len(imp)), imp.index)
plt.title('Feature Importance', fontsize=14)
plt.show()
```





4.2 Regression

Define the Model

In [32]:

```
# define model
rf = RandomForestRegressor(random_state=RSEED)

# define parameter grid
parameters_grid = {
    'max_depth': [3, 5],
    'min_samples_leaf': [2, 8],
    'n_estimators': [20, 50],
    'max_features': [2, 4]
}

# define grid search
grid_search = GridSearchCV(estimator=rf, param_grid=parameters_grid, cv=10)
```

Fit the Model

In [33]:

```
# fit estimator
grid_search.fit(X_reg_train, y_reg_train)

# get best estimator
best = grid_search.best_estimator_
```

In [34]:

```
# print best parameters
pd.DataFrame.from_dict(grid_search.best_params_, orient='index', columns=['Selected Value']).T
```

Out[34]:

	max_depth	max_features	min_samples_leaf	n_estimators
Selected Value	5	2	8	50

Predict

In [35]:

```
# predict
y_pred = best.predict(X_reg_test)
```

Evaluate the Model

In [36]:

```
# calculate MSE
MSE = round(mean_squared_error(y_reg_test, y_pred), 3)

df = pd.DataFrame([MSE]).T
df = df.rename(index={0: 'Random Forest Regressor'}, columns={0: 'MSE'})
df
```

Out[36]:

	MSE
Random Forest Regressor	149.387

Feature Importance

In [37]:

```
# get feature importance
imp = pd.DataFrame(grid_search.best_estimator_.fit(X_reg_train, y_reg_train).feature_importances_,
                   index=X_reg_train.columns, columns=['Importance']).sort_values('Importance')
```

In [38]:

```
# plot feature importance
plt.figure(figsize=(8, 6))
plt.barh(range(0, len(imp)), imp['Importance'], color='#9ecae1')
plt.grid(axis='x', alpha=0.5, color='lightgrey')
plt.yticks(range(0, len(imp)), imp.index)
plt.title('Feature Importance', fontsize=14)
plt.show()
```