In the final exercise of the **Intro to Machine Learning** course, you learned how to make a submission to a Kaggle competition. But some of the work was already completed for you, since you were provided a notebook with partially completed code.

In this tutorial, you'll explore a **full workflow** that you can use to get started (from the very beginning!) with creating a submission to any Kaggle competition. We'll use the **Titanic competition** as an example.

Part 1: Get started

In this section, you'll learn more about the competition and make your first submission.

Join the competition!

The first thing to do is to join the competition! Open a new window with **the competition page**, and click on the **"Join Competition"** button, if you haven't already. (*If you see a "Submit Predictions" button instead of a "Join Competition" button, you have already joined the competition, and don't need to do so again.*)

This takes you to the rules acceptance page. You must accept the competition rules in order to participate. These rules govern how many submissions you can make per day, the maximum team size, and other competition-specific details. Then, click on **"I Understand and Accept"** to indicate that you will abide by the competition rules.

The challenge

The competition is simple: we want you to use the Titanic passenger data (name, age, price of ticket, etc) to try to predict who will survive and who will die.

The data

To take a look at the competition data, click on the **Data tab** at the top of the competition page. Then, scroll down to find the list of files.

There are three files in the data: (1) **train.csv**, (2) **test.csv**, and (3) **gender_submission.csv.**

(1) train.csv

train.csv contains the details of a subset of the passengers on board (891 passengers, to be exact -- where each passenger gets a different row in the table). To investigate this data, click on the name of the file under the **"Data Sources"** column (on the left of the screen). Once you've done this, all of the column names (along with a brief description of what they contain) are listed to the right of the screen, under the **"Columns"** heading.

You can view all of the data in the same window.

The values in the second column (**"Survived"**) can be used to determine whether each passenger survived or not:

- if it's a "1", the passenger survived.
- if it's a "0", the passenger died.

For instance, the first passenger listed in **train.csv** is Mr. Owen Harris Braund. He was 22 years old when he died on the Titanic.

(2) test.csv

Using the patterns you find in **train.csv**, you have to predict whether the other 418 passengers on board (in **test.csv**) survived.

Click on **test.csv** (under the **"Data Sources"** column) to examine its contents. Note that **test.csv** does not have a **"Survived"** column - this information is hidden from you, and how well you do at predicting these hidden values will determine how highly you score in the competition!

(3) gender_submission.csv

The **gender_submission.csv** file is provided as an example that shows how you should structure your predictions. It predicts that all female passengers survived, and all male passengers died. Your hypotheses regarding survival will probably be different, which will lead to a different submission file. But, just like this file, your submission should have:

- a **"PassengerId"** column containing the IDs of each passenger from **test.csv.**
- a **"Survived"** column (that you will create!) with a "1" for the rows where you think the passenger survived, and a "0" where you predict that the passenger died.

Your first submission

As a benchmark, you'll download the **gender_submission.csv** file and submit it to the competition. Begin by clicking on the download link to the right of the name of the file.

This downloads the file to your computer. Then:

- Click on the blue **"Submit Predictions"** button in the top right corner of the competition page. (*This button now appears where the **"Join Competition"** button was.*)
- Scroll down to **"Step 1: Upload submission file"**. Upload the file you just downloaded. Then, click on the blue **"Make Submission"** button.

In a few seconds, your submission will be scored, and you'll receive a spot on the leaderboard. Next, we'll walk you through how to outperform this initial submission!

Part 2: Your coding environment

In this section, you'll train your own machine learning model to improve your predictions.

The Notebook

The first thing to do is to create a Kaggle Notebook where you'll store all of your code. You can use Kaggle Notebooks to getting up and running with writing code quickly, and without having to install anything on your computer. (*If you are interested in deep learning, we also offer free GPU and TPU access!*)

Begin by clicking on the **Notebooks tab** on the competition page. Then, click on **"New Notebook".**

Next, click on **"Create"**. (*Don't change the default settings: so, **"Python"** should appear under "Select language", and you should have **"Notebook"** selected under "Select type".*)

Your notebook will take a few seconds to load. In the top left corner, you can see the name of your notebook -- something like **"kernel2daed3cd79".**

You can edit the name by clicking on it. Change it to something more descriptive, like **"Getting Started with Titanic".**

Your first lines of code

When you start a new notebook, it has two gray boxes for storing code. We refer to these gray boxes as "code cells".

The first code cell already has some code in it. To run this code, put your cursor in the code cell. (*If your cursor is in the right place, you'll notice a blue vertical line to the left of the gray box.*) Then, either hit the play button (which appears to the left of the blue line), or hit **[Shift] + [Enter]** on your keyboard.

If the code runs successfully, three lines of output are returned. Below, you can see the same code that you just ran, along with the output that you should see in your notebook.

In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# Any results you write to the current directory are saved as output.
```

```
/kaggle/input/titanic/gender_submission.csv
/kaggle/input/titanic/test.csv
/kaggle/input/titanic/train.csv
/kaggle/input/melbourne-housing-snapshot/melb_data.csv
```

This shows us where the competition data is stored, so that we can load the files into the notebook. We'll do that next.

Load the data

The second code cell in your notebook now appears below the three lines of output with the file locations.

Type the two lines of code below into your second code cell. Then, once you're done, either click on the blue play button, or hit **[Shift] + [Enter]**.

In [2]:

```
train_data = pd.read_csv("../input/titanic/train.csv")
train_data.head()
```

Out[2]:

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

Your code should return the output above, which corresponds to the first five rows of the table in **train.csv**. It's very important that you see this output **in your notebook** before proceeding with the tutorial!

> *If your code does not produce this output*, double-check that your code is identical to the two lines above. And, make sure your cursor is in the code cell before hitting **[Shift] + [Enter].**

The code that you've just written is in the Python programming language. It uses a Python "module" called **pandas** (abbreviated as `pd`) to load the table from the **train.csv** file into the notebook. To do this, we needed to plug in the location of the file (which we saw was `/kaggle/input/titanic/train.csv`).

> If you're not already familiar with Python (and pandas), the code shouldn't make sense to you -- but don't worry! The point of this tutorial is to (quickly!) make your first submission to the competition. At the end of the tutorial, we suggest resources to continue your learning.

At this point, you should have at least three code cells in your notebook.

Copy the code below into the third code cell of your notebook to load the contents of the **test.csv** file. Don't forget to click on the play button (or hit **[Shift] + [Enter]**)!

In [3]:
```
test_data = pd.read_csv("../input/titanic/test.csv")
test_data.head()
```

Out[3]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Em
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

As before, make sure that you see the output above in your notebook before continuing.

Once all of the code runs successfully, all of the data (in **train.csv** and **test.csv**) is loaded in the notebook. (*The code above shows only the first 5 rows of each table, but all of the data is there -- all 891 rows of **train.csv** and all 418 rows of **test.csv**!*)

Part 3: Improve your score

Remember our goal: we want to find patterns in **train.csv** that help us predict whether the passengers in **test.csv** survived.

It might initially feel overwhelming to look for patterns, when there's so much data to sort through. So, we'll start simple.

Explore a pattern

Remember that the sample submission file in **gender_submission.csv** assumes that all female passengers survived (and all male passengers died).

Is this a reasonable first guess? We'll check if this pattern holds true in the data (in **train.csv**).

Copy the code below into a new code cell. Then, run the cell.

In [4]:
```
women = train_data.loc[train_data.Sex == 'female']["Survived"]
rate_women = sum(women)/len(women)

print("% of women who survived:", rate_women)
```

```
% of women who survived: 0.7420382165605095
```

Before moving on, make sure that your code returns the output above. The code above calculates the percentage of female passengers (in **train.csv**) who survived.

Then, run the code below in another code cell:

In [5]:

```
men = train_data.loc[train_data.Sex == 'male']["Survived"]
rate_men = sum(men)/len(men)

print("% of men who survived:", rate_men)
```

```
% of men who survived: 0.18890814558058924
```

The code above calculates the percentage of male passengers (in **train.csv**) who survived.

From this you can see that almost 75% of the women on board survived, whereas only 19% of the men lived to tell about it. Since gender seems to be such a strong indicator of survival, the submission file in **gender_submission.csv** is not a bad first guess, and it makes sense that it performed reasonably well!

But at the end of the day, this gender-based submission bases its predictions on only a single column. As you can imagine, by considering multiple columns, we can discover more complex patterns that can potentially yield better-informed predictions. Since it is quite difficult to consider several columns at once (or, it would take a long time to consider all possible patterns in many different columns simultaneously), we'll use machine learning to automate this for us.

Your first machine learning model

We'll build a **random forest model**. This model is constructed of several "trees" (there are three trees in the picture below, but we'll construct 100!) that will individually consider each passenger's data and vote on whether the individual survived. Then, the random forest model makes a democratic decision: the outcome with the most votes wins!

The code cell below looks for patterns in four different columns (**"Pclass"**, **"Sex"**, **"SibSp"**, and **"Parch"**) of the data. It constructs the trees in the random forest model based on patterns in the **train.csv** file, before generating predictions for the passengers in **test.csv**. The code also saves these new predictions in a CSV file **my_submission.csv.**

Copy this code into your notebook, and run it in a new code cell.

In [6]:

```
from sklearn.ensemble import RandomForestClassifier

y = train_data["Survived"]

features = ["Pclass", "Sex", "SibSp", "Parch"]
X = pd.get_dummies(train_data[features])
X_test = pd.get_dummies(test_data[features])

model = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=1)
model.fit(X, y)
predictions = model.predict(X_test)

output = pd.DataFrame({'PassengerId': test_data.PassengerId, 'Survived': predicti
ons})
output.to_csv('my_submission.csv', index=False)
print("Your submission was successfully saved!")
```

```
Your submission was successfully saved!
```

Make sure that your notebook outputs the same message above (`Your submission was successfully saved!`) before moving on.

> Again, don't worry if this code doesn't make sense to you! For now, we'll focus on how to generate and submit predictions.

Once you're ready, click on the blue **"Save Version"** button in the top right corner of your notebook. This will generate a pop-up window.

- Ensure that the **"Save and Run All"** option is selected, and then click on the blue **"Save"** button.
- This generates a window in the bottom left corner of the notebook. After it has finished running, click on the number to the right of the **"Save Version"** button. This pulls up a list of versions on the right of the screen. Click on the ellipsis **(...)** to the right of the most recent version, and select **Open in Viewer.**
- Click on the **Output** tab on the right of the screen. Then, click on the **"Submit to Competition"** button to submit your results.

Once your file is successfully submitted, you should receive a message saying that you've moved up the leaderboard. Great work!

Part 4: Keep learning!

Can you use what you learned about random forests in the **Intro to Machine Learning** course to generate even better predictions?

Check out the **Intermediate Machine Learning** course to learn about more advanced techniques!