If your computer has limited amount of memory or just if you want to save time when experimenting, you can work on a subgraph of the airport dataset, for instance only with the most important nodes, or only nodes in a region of the world.

1. Training and Validation set
   
   (a) Let’s start by creating a training set for the Airport dataset. A training set is composed of $t$ edges (taken at random) and $t$ pairs of nodes without edges (taken at random). You can use random.sample (https://www.geeksforgeeks.org/python-random-sample-function/) to pick randomly edges (or pairs of nodes). Typically, you can choose $t = L/6$ ($L$ is the number of edges in the original graph). Keep randomly chosen edges and non-edges in separated lists. Note that, when the graph is large, creating the list of all pair of nodes without edges is costly. Instead, you can select 2 list of nodes with repetitions of the same size, and then consider that to each position of the lists corresponds a pair of nodes. Don’t forget to remove loops, duplicates and actual edges.

   (b) Remove edges of the training set from the graph.

2. Computing heuristics
   
   (a) Using existing functions in networkx.(adamic_adar_index, etc., see https://networkx.org/documentation/stable/reference/algorithms/link_prediction.html) compute common heuristics between all (or a sample of) pairs of nodes on the graph.

   (b) Find the 20 node pairs of higher and lower scores, for each heuristic. Are these rankings intuitively a good starting point? (A simple way to sort is to transform the output of heuristics into dictionaries (dict(nx.adamic_adar_index(...))), and then use the same method as in previous experiments (or search for something like "sort dictionary by value python" in google.)

3. Using Machine Learning
   
   (a) We will use the sklearn.linear_model.LogisticRegression function to train our model. To train the model, we will use the following method: clf = LogisticRegression().fit(X, y), as in the example of the documentation. We need to prepare $X$ and $y$. $X$ represents the input and can be provided as a list of list: each of the internal list corresponds to the features of one node pair. $y$ is the list of values to predict. For instance, $X=[[1,3,1],[2,20,10]]$, $y=[1,0]$ corresponds to a training set with 2 examples, each having 3 features, the first one being an edge and the second one a non-edge. Prepare $X$ (combining all computed heuristics) and $y$ from the training set. (you can create $y$ by using something like [1]*len(train_edges)+[0]*len(train_non_edges) . For $X$, you code can look something like [AA[e],CN[e],PA[e]] for e in train_edges+train_non_edges]

   (b) Train the model. (call fit(X,y))

   (c) Use function LogisticRegression.predict_proba(Xvalidate) to get predictions for all pairs of nodes without edges in the original graph.

   (d) Sort the most likely pairs of nodes. Check the results.

4. Going further: Node attributes prediction
   
   (a) Hide the country information of 20% of airports. We could imagine that this information was missing in the database. Propose a ML based method to assign a country to those airports and check the results.