

Test #3

Guidelines:

- Open-book test, no collaboration with others allowed during the test.
- For each of the following 3 problems, please submit a Jupyter notebook that do things as described in the problem set.

#1. Information gain: Reuse the Jupyter notebook you developed for **Lab #5 Analysis of information gain from rock-paper-scissor transcripts using numpy** to analyze RPS_transcriptX.txt, RPS_transcriptY.txt, and RPS_transcriptZ.txt.

Please do the following:

- Analyze RPS_transcriptX.txt, RPS_transcriptY.txt, RPS_transcriptZ.txt:
These three files enclosed in the folder use the same format like that for RPS_transcript1.txt to record 1000 matches between the user and 3 new agents X, Y, and Z respectively. Note that they each records 1000 matches, not 300 matches in RPS_transcript1.txt. For each of the agents X, Y, and Z respectively, please do things similar to what you did in **Lab #5** to find out the information gains provided by each of the features (i.e. actions of the user and the agent in the previous two matches) toward the prediction of the agent's action now.
- Recording your findings:
Based on the findings above, for each of the agents X, Y, and Z, please record (i) the single most important feature that leads to maximal information gain and (ii) the amount of such information gain from the feature. Note that these correspond to the most important features for predicting the behavior of the agents. Record your answers as comments in the very last cell of your notebook. Upload your notebook.

#2 Rock-Paper-Scissor and classification algorithms in Scikitlean:

Please carefully examine the contents of Test3_RockPaperScissor_Scikeitlearn.ipynb enclosed in this folder to see how you may use Scikitlean to apply Naïve Bayes to the Rock-Paper-Scissor transcripts for Agent #1 and Agent #2 as shown in RPS_transcript1.txt and RPS_transcript2.txt. In each case, it sets up the data into a dataframe object, split the data into a training subset and a testing subset, apply Naïve Bayes to learn from the training subset and test it on the testing subset to see the accuracy of prediction. Note that the steps in Cell [1] to [17] learn and predict the action now for each agent in two settings: (i) using the actions of the two sides in the previous two matches as features or (ii) using the actions of the two sides in the previous match as features only.

Please do the following:

- a) Open the Jupyter notebook **Test3_RockPaperScissor_Scikitlearn.ipynb** in the folder. Go through and understand all the steps shown in Cell [1] to [17] in order to use Scikitlearn to conduct Naïve Bayes.
- b) Apply Naïve Bayes use Scikitlearn to learn from RPS_transcriptX.txt, RPS_transcriptY.txt, RPS_transcriptZ.txt respectively:
Again these three files enclosed in the folder use the same format like that for RPS_transcript1.txt to record 1000 matches between the user and 3 new agents X, Y, and Z respectively. Note that they each records 1000 matches, not 300 matches in RPS_transcript1.txt. In cells after [17], please do things similar to the steps shown in Cell [1] to [17] to find out **the accuracy of prediction** using the same Naïve Bayes approach. Note that the steps in Cell [1] to [17] learn and predict the action now for each agent in two settings: (i) using the actions of the two sides in the previous two matches as features or (ii) using the actions of the two sides in the previous match as features only.
- c) Recording your findings:
Based on the findings above, for each of the agents X, Y, and Z, please record **the accuracy of prediction** in two settings: (i) based on the actions of the two sides in the previous two matches as features or (ii) based on the actions of the two sides in the previous match as features only. Record your answers as comments in the very last cell of your notebook. Upload your notebook.

#3. Linear regression, kernel trick, and overfitting:

Please carefully examine the contents of Test3_LinearRegression.ipynb enclosed in this folder to see how you may use Scikitlearn to apply linear regression to learn polynomial functions of degree 1, 2, and 4, respectively. In each case, it applies linear regression to learn from the noisy training data points. Then we check (i) the sum of squared errors between the predicted y values and the **noisy y values** of the training data points versus (ii) the sum of squared errors between the predicted y values and the **true y values**.

Please do the following:

- a) Open the notebook Test3_LinearRegression.ipynb in the folder. Go through and understand all the steps shown in Cell [1] to [13].
- b) Learn a polynomial function of degree 5 using linear regression:
After cell [13], repeat the same linear regression approach shown in Cell [1] to [13] to continue the experiments of applying linear regression to learn polynomial functions of degree 5 and 6 respectively from the noisy training data points. And check (i) the sum of squared errors between the predicted y values and the **noisy y values** of the training data points versus (ii) the sum of squared errors between the predicted y values and the **true y values**.
- c) Recording your findings:
Record the sums of squared errors you found above in a cell in the end. Do you keep getting a smaller sum of squared errors between the predicted y values and the **noisy y values** of the

training data points? Do you keep getting a larger sum of squared errors between the predicted y values and the **true y values**? If so, you have produced the phenomenon of overfitting here. This shows that using a highly complex model that fits the training data very well may go too far and does not truly reflect the true relationship behind the scene.