Computational Astrophysics 2023/2024

Physics of Data

Part-2: *Exoplanet Detector*

Submission deadline: 23/12/2024

Assignment 2: The Second Renaissance

““The Three Laws of Robotics:

1: A robot may not injure a human being or, through inaction, allow a human being to come to harm;

2: A robot must obey the orders given it by human beings except where such orders would conflict with the First Law;

3: A robot must protect its own existence as long as such protection does not conflict with the First or Second Law;

The Zeroth Law: A robot may not harm humanity, or, by inaction, allow humanity to come to harm.”

”

Isaac Asimov - I, Robot

Learning aims: Through the solution of this exercise, you will generate your exoplanetary transit light curve with the batman package. You will familiarise yourself with the transit shapes and detect them using Machine Learning and Deep Learning algorithms.

Tasks:

A. **A transit to conquer them all:** Generate a transit light curve using the Batman tutorial. Choose a real known planet from https://exoplanet.eu and pick all the suitable parameters for both the star and the planet. Use the correct limb darkening parameters for the star you chose using the Limb Darkening Calculator. Generate a Figure, name all the axes with the correct unit measurements and display the name of the planet in the title. Push the figure on your GitHub repository with the name "**assignment2_taskA.png**". Report the *(If you completed Task F in assignment 1, this will be almost automatic. In the submission report, the date and hashcode of the commit associated with the creation of the file).*

B. **Planetary brothers:** Add a second transiting planet to your previous target, use the same star and decrease the radius of the planet by a factor of ½. Generate a Figure with two planets, showing both components with two different colours, and use a legend to distinguish them. Push the figure on your GitHub repository with the name "**assignment2_taskB.png**" *(In the submission, report the date and hashcode of the commit associated with the creation of the file assignment2_taskB.txt.)*

B. **We are family now:** Add a third transiting planet to your target, use the same star and increase the radius of the planet by a factor of 2. Generate a Figure with two planets, showing the three components with three different colours. Use a legend to distinguish them. Push the figure on your GitHub repository with the name "**assignment2_taskC.png**" *(In the submission, report the date and hashcode of the commit associated with the creation of the file assignment2_taskC.txt.)*

D. **Kepler detection:** Use the notebook "**notebook_SVM.ipynb**" to detect real planets in the '**kepler/data_no_injection'** dataset using Support Vector Machines, use three different kernels 'linear', 'gaussian', and 'polynomial' of degree 4 to vet the original dataset without injected planets. Display the confusion matrix and the precision for all the kernels in a file called "**report_no_injection_assignment2_taskD.txt**". Hint: Use the SVC model of scikit-learn, and go to the documentation to see how to use the three kernels. *(In the submission, report the date and hashcode of the commit associated with the creation of the file report_no_injection_assignment2_taskD.txt.).*

E. **Kepler augmented detection!:** Use the notebook "**notebook_SVM.ipynb"** to detect real planets in the '**kepler/data_injected'** dataset using Support Vector Machines, use three different kernels 'linear', 'gaussian', and 'polynomial' of degree 4 to vet the original dataset without injected planets. Display the confusion matrix and the precision for all the kernels in a file called "**report_injection_assignment2_taskE.txt**". Hint: Use the SVC model of scikit-learn, and go to the documentation to use the three kernels. *(In the submission, report the date and hashcode of the commit associated with the creation of the file report_injection_assignment2_taskE.txt.).*

F. **Kepler mind!:** Use the notebook "**notebook_NN.ipynb"** to detect real planets in the '**kepler/data_no_injection'** dataset using Artificial Neural Networks, use the default layers and try to add other layers to the network to vet the original dataset without injected planets. The number of layers and the neurons for all the hidden layers are arbitrary, please add a Dropout layer and use the 'relu' activation function after each hidden layer. Display the confusion matrix and the precision for all the networks in a file called "**report_no_injection_assignment2_taskE.txt**". Hint: Use the Tensorflow tutorial of Tensorflow to see the correct usage. *(In the submission, report the date and hashcode of the commit associated with the creation of the file report_no_injection_assignment2_taskF.txt.).*

G. **Kepler changed my mind!:** Use the notebook "**notebook_NN.ipynb"** to detect real planets in the '**kepler/data_injected'** dataset using Artificial Neural Networks, use the default layers and try to add other layers to the network to vet the original dataset without injected planets. The number of layers and the neurons for all the hidden layers are arbitrary, please add a Dropout layer and use the 'relu' activation function after each hidden layer. Display the confusion matrix and the precision for all the networks in a file called "**report_injected_assignment2_taskG.txt**". Hint: Use the Tensorflow tutorial of Tensorflow to see the correct usage. *(In the submission report the* *date and hashcode of the commit associated with the creation of the file report_injected_assignment2_taskG.txt.).*

Challenges (not mandatory):

H. **One command to support them:** Define in the daneel.detection folder a class to detect exoplanets using SVMs. This class should be called from the command line (with the flag -d svm or –detection svm). Describe how to call this method in the README file. Define the kernel and the dataset path in the parameters.yaml file. Finally, commit and push the changes to the main branch. The command should look like:
daneel -i path_to_the_parameters.yaml -d svm

This command should plot the statistics of the SVM as in the "**notebook_SVM.ipynb"** notebook

I. **One command to read their mind:** Define in the daneel.detection folder a class to detect exoplanets using NNs. This class should be called from the command line (with the flag -d nn or –detection nn). Describe how to call this method in the README file. Define the kernel and the dataset path in the parameters.yaml file. Finally, commit and push the changes to the main branch. The command should look like:
daneel -i path_to_the_parameters.yaml -d nn

This command should plot the statistics of the SVM as in the "**notebook_NN.ipynb"** notebook

I. **One command to find complex pattern:** Define in the daneel.detection folder a class to detect exoplanets using CNNs. This class should be called from the command line (with the flag -d cnn or –detection cnn). Describe how to call this method in the README file. Define the kernel and the dataset path in the parameters.yaml file. Finally, commit and push the changes to the main branch. The command should look like:
daneel -i path_to_the_parameters.yaml -d cnn

Hint: see the Tensorflow tutorial to see how CNN work. I suggest preprocess the dataset and re-create the light curves in the following way:

a) Cut the flux in a shape of dimension 3136 (from 3197) to have a perfect square;
b) Reshape each element in a square of dimension 56x56;
c) Apply the usual kernel and Conv2D layers within the CNN

J) **Dreaming new worlds:** Once you created the dataset where each lightcurve is a square object (see previous task), train a Generative adversarial network to generate new lightcurves. Hint: Use the GAN defined in the "**Machine_learning_theoretical_lecture.ipynb**" notebook. Create a daneel.dream folder within which you will define the GAN architecture that allows your Python package to "dream" a new exoplanetary transit light curve.

The daneel package should be able to imagine them with the command daneel -i path_to_the_parameters.yaml --dream

You will point the dataset to replicate within the parameters.yaml file.

For each of these commits, report the commit hash and the task of the assignments.