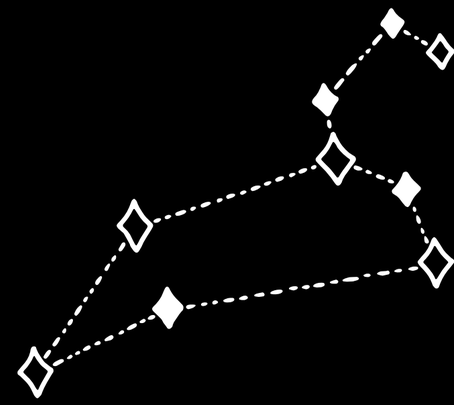
WEB-BASED AUTOMATIC CLASSIFICATION OF EXOPLANETS

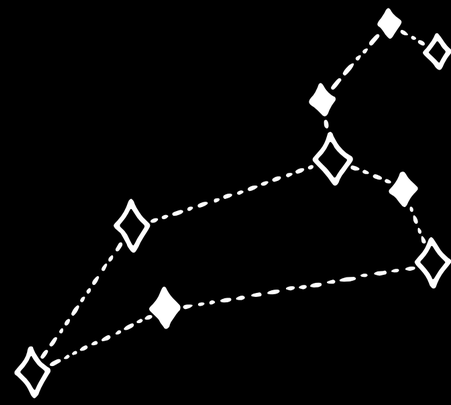


Team Name: ASTROPHEL

Abhishek mittal 200030003
Devesh kumar 200030017
Kamtekar varad 200030025
Sanghasheela 200030050
Shrishti 200030052
Siddhant 200030054

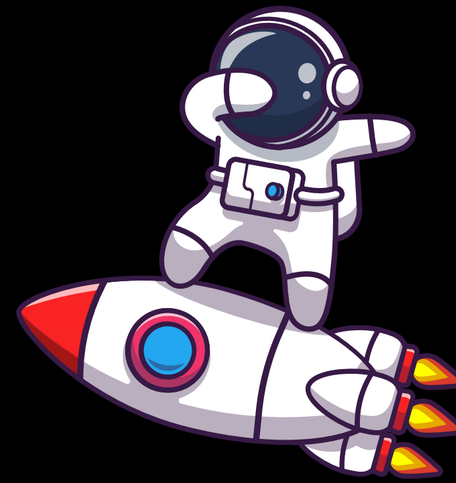
Problem Statement

Build a stand-alone web-based application using open-source software(s) to identify and categorize The Threshold crossing events (TCE's). From the data given TCE's must be classified into different classes. False detection should be minimum, and TCE's must be Detected.



The problem can be broadly categorized into two parts:

1. Developing a statistical/machine learning model to cover the mentioned parameters.
2. Deploying it efficiently to a stand-alone application and web-based tool (no additional APIs used).



Build with

Python

- sklearn
- seaborn
- pandas
- csv

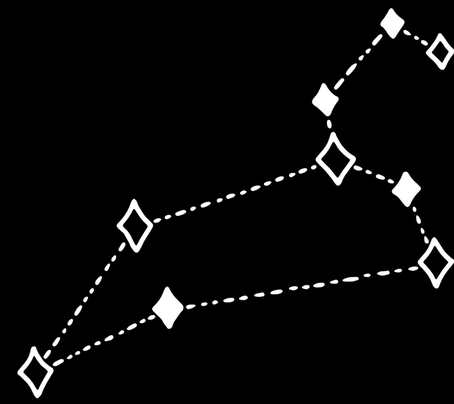
Node.JS

- PythonShell

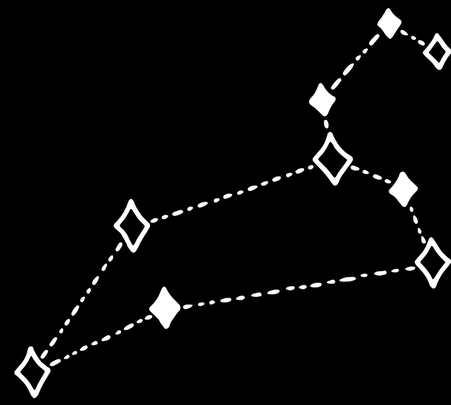
Express JS

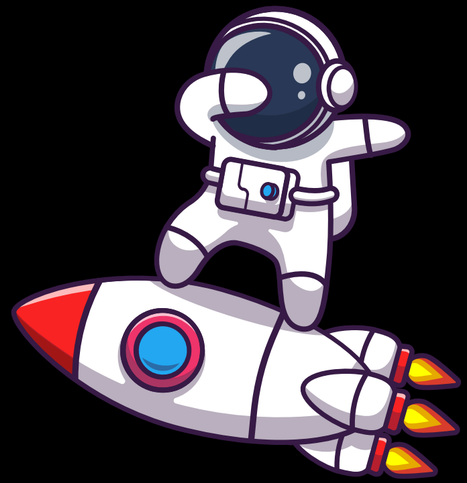
HTML

CSS



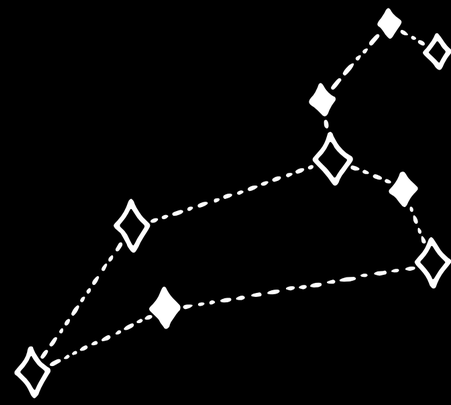
Machine Learning Model



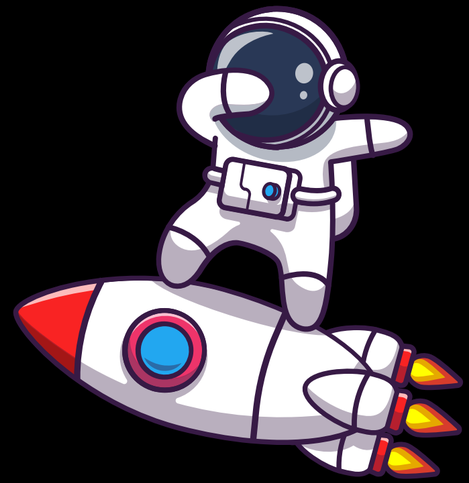


Data

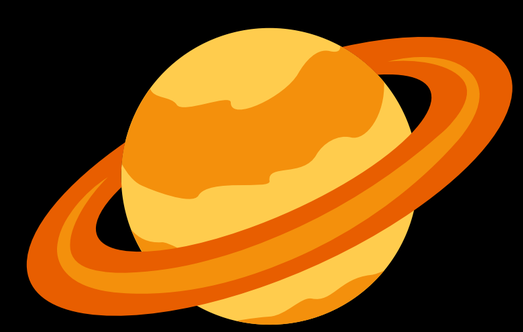
For training
Keplars TCE dataset (In csv format) produced by the
NASA Exoplanet Archive
http://exoplanetarchive.ipac.caltech.edu



Input data – Should be a CSV file conatining values of the features used and in the given order
'tce_period', 'tce_timeObk_err', 'tce_impact_err', 'tce_depth', 'tce_depth_err', 'tce_prad_err', 'tce_steff_err', 'tce_slogg_err'



Feature selected

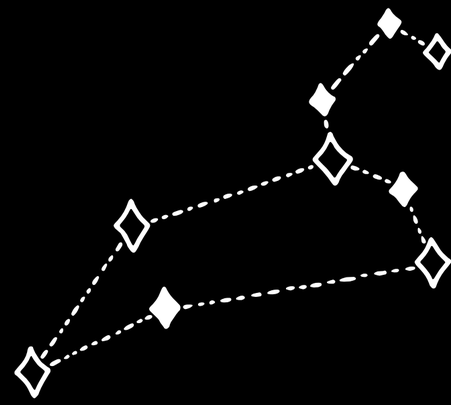


Done by SelectKBest Function of Sklearn

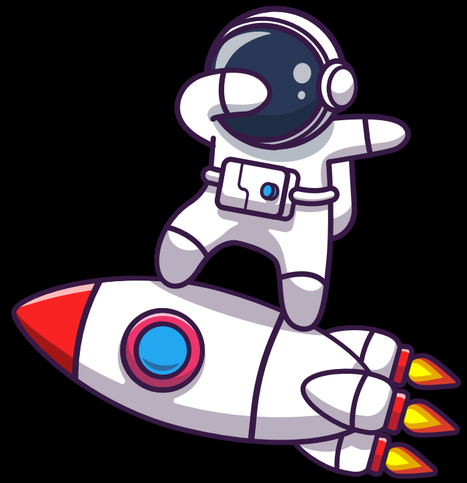
Varied the features from 2-21

the accuracy achieved at 8 features and onwards was nearly same or lower

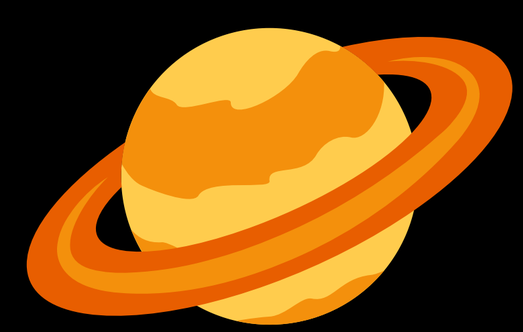
Result - Using 8 features



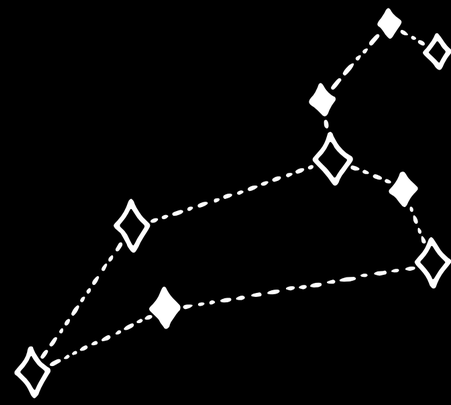
'tce_period', 'tce_timeObk_err', 'tce_impact_err', 'tce_depth', 'tce_depth_err', 'tce_prad_err', 'tce_steff_err', 'tce_slogg_err'

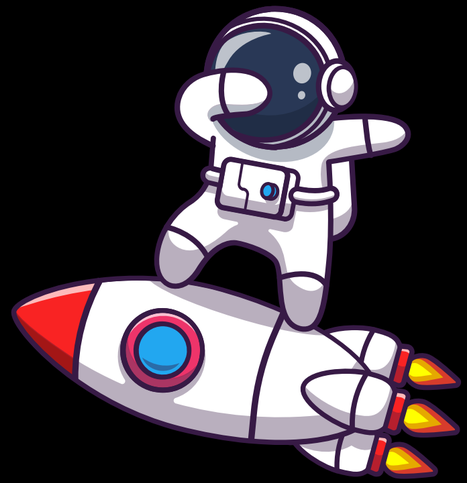


Model used

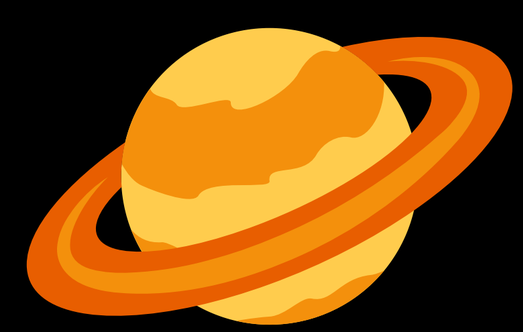


Using Score probability combination of Random forest with 100 trees and k-nearest neighbour while using the 8 features from the given list
As this model was performing better than the other models tried SVM,LR,SVC,Neural Network, GCP(slow) in terms of speed and accuracy

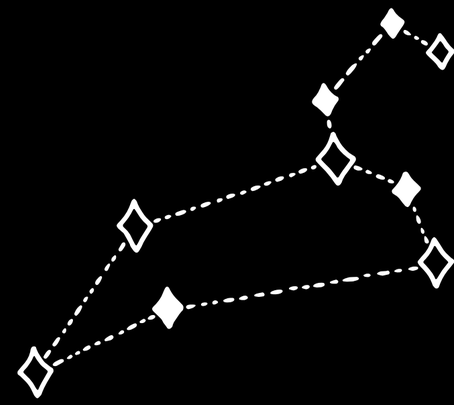




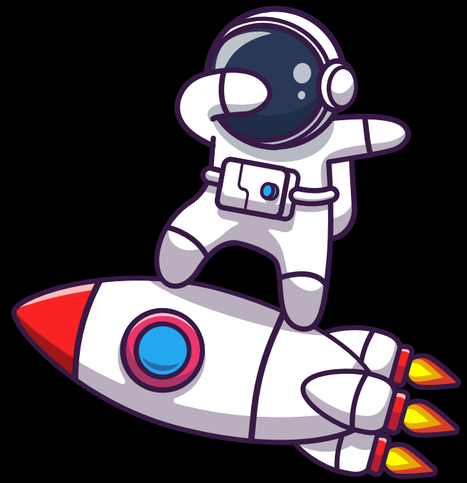
Results



Accuracy
63.5% on the 1000 test cases
(spliting the Keplars Dataset in 14000 training and 1000 test)
runtime ~6 seconds



Can be checked by running the accu function at the end of python file which would show accuracy and a confusion matrix using the seaborn function heatmap.

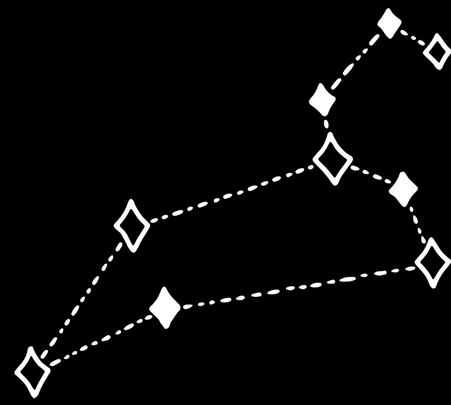


Output Format

- A CSV file conatining one column with name "CLASS" containing the predicted TCE type of that row.

Features.txt file containing

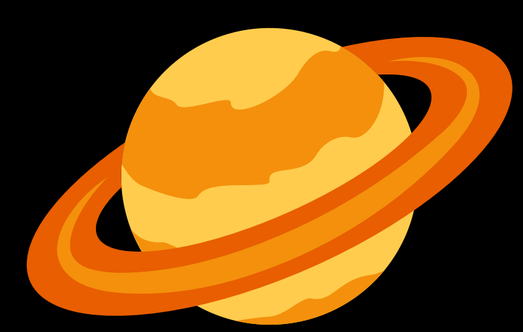
- Count of the different classes in the output
- The features used by the model



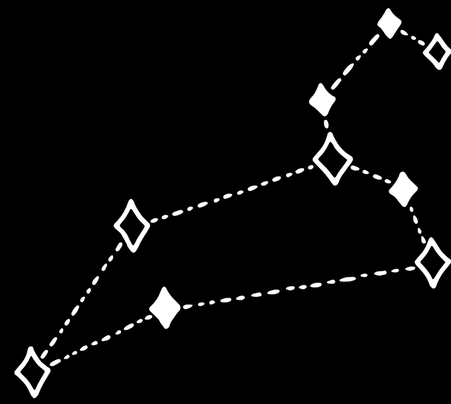
On WebPage

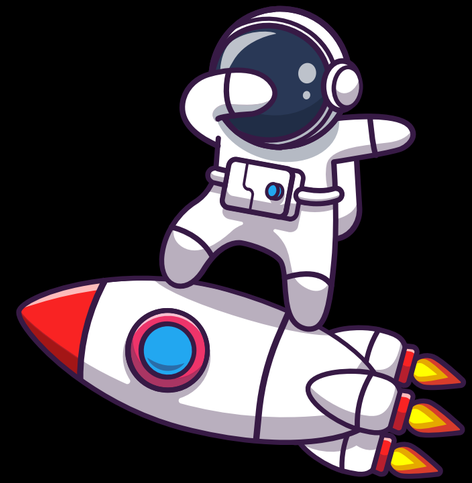
- Link to download to resultant zip file containg CSV and features.txt

Drawbacks



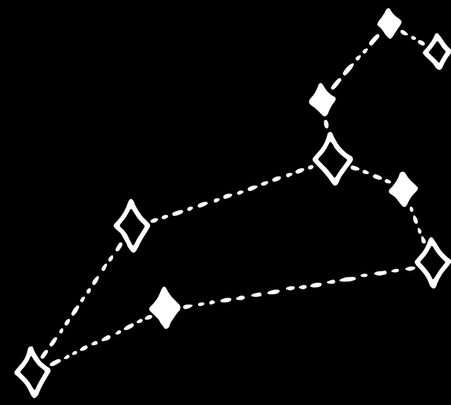
- The features have to be in a given order in the input CSV file
- In case of equal probability Preference would be AFP>PC>NTP>UNK
- Accuracy can be improved

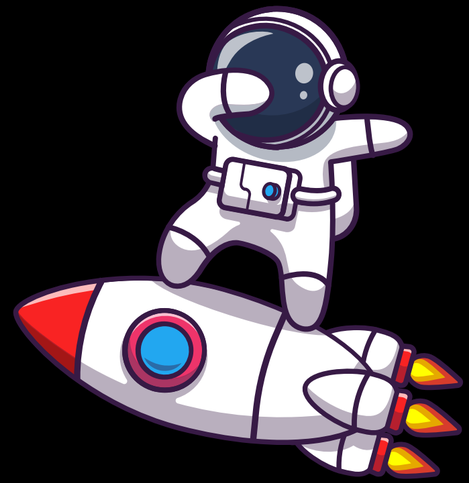




FRONTEND

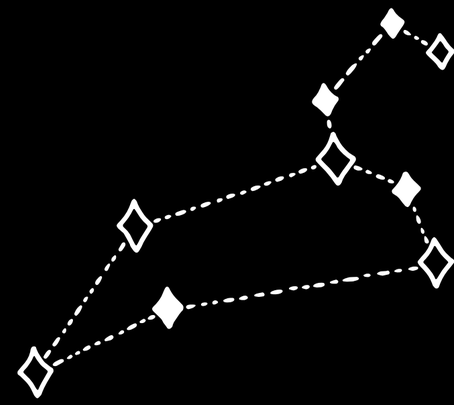
- Two input fields : Select input CSV File , save
- On the next page, button to download zip file (output CSV file and a txt file containing count of each class and features list.

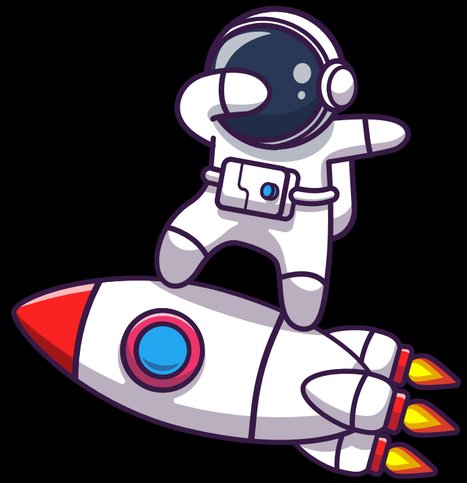




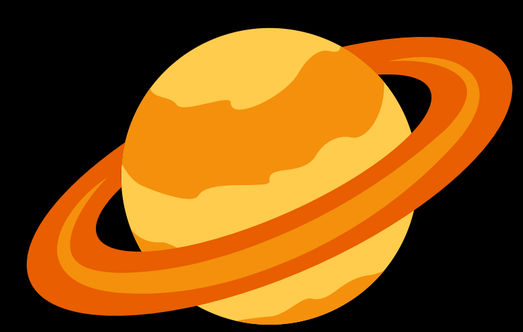
BACKEND

- Take input from webpage
- Store it in upload folder
- Run model
- Print ouput (list of class, feature list, output csv file)

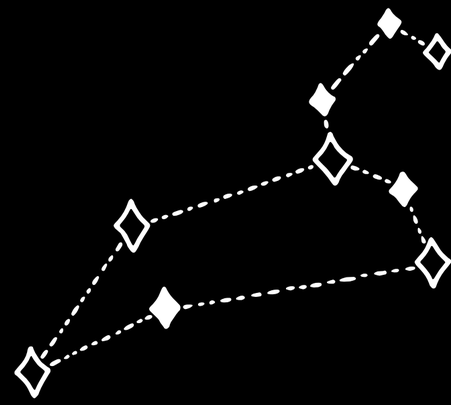




Future Developments

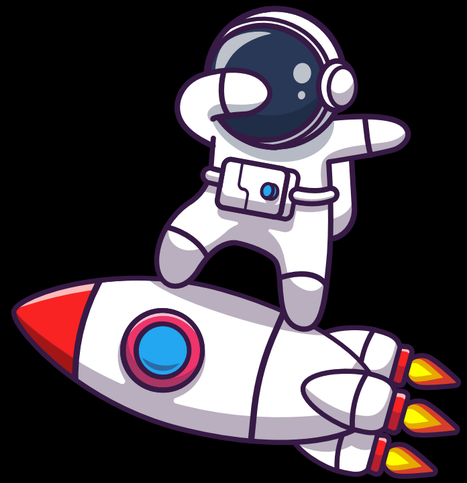


- In case of equal probaibility find a better method for resolving it and providing a more accuracte classification.



- Model should run irrespective of the order of features in CSV file.

- Try other models or features which can give improved accuracy time efficiently



Thank you