Transiting Planet Discovery in the Kepler Pipeline Using Automated Machine Learning - Code Walkthrough

Martin Mohan

Research in Computing Presentation
MSC in Data Analytics - 2020
National College of Ireland

x18191339@student.ncirl.ie

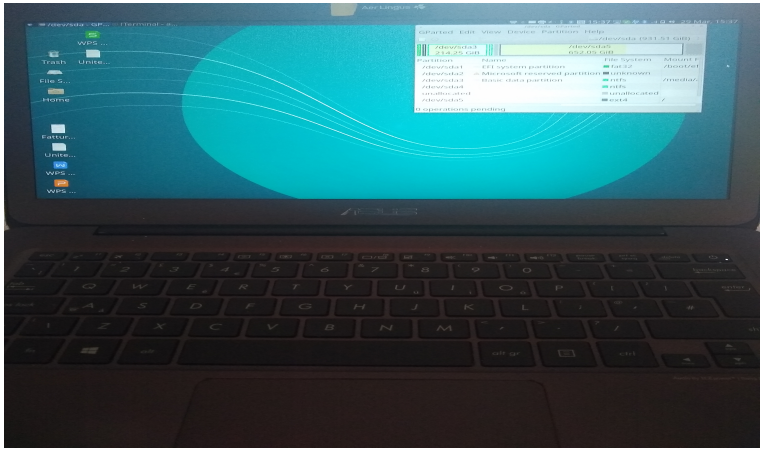
August 17, 2020

Overview

1. Development Environment
2. SEMMA pipeline overview
3. Treat 1,2,3,4,5
4. Results
5. Unit Tests
6. All treatments object orientated using __doc__ for documentation

Gaming desktop and a laptop running on Ubuntu

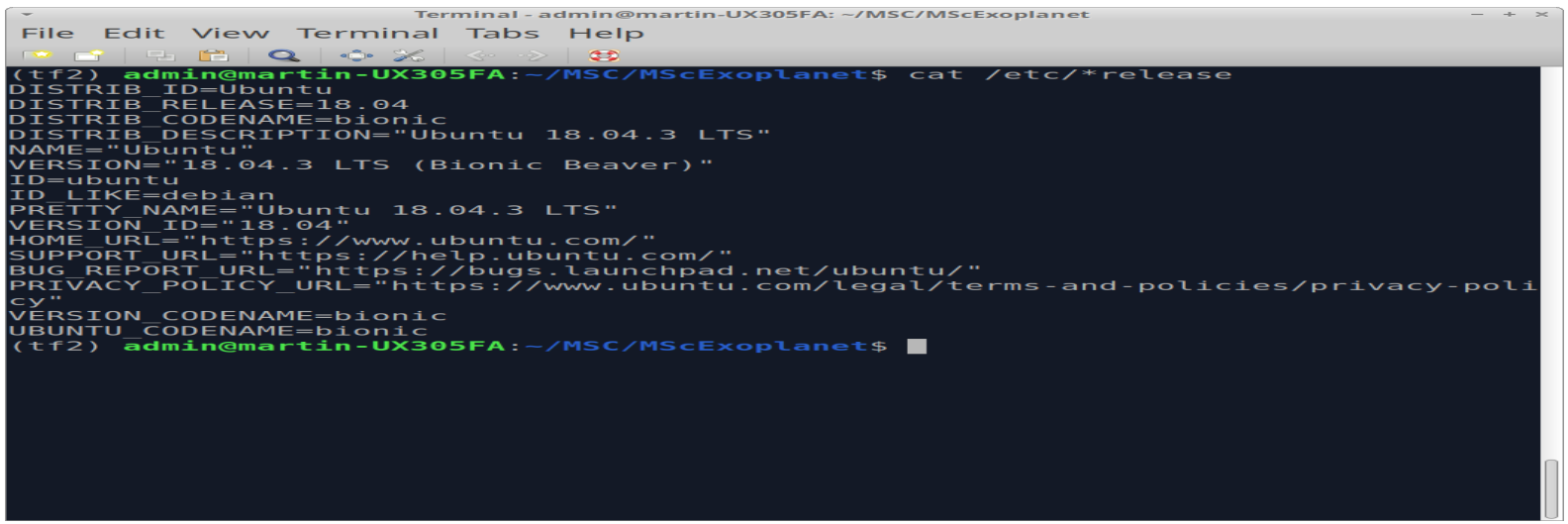
(a) HP - OMEN 875-0051 Gaming PC
2 TB HDD / 256 GB SSD disk
Intel® CoreTM i7, RTX 2060



(b) Asus Zenbook UX30FA laptop
1TB SSD disk
Intel® 5Y10 CPU @ 0.80GHz

Ubuntu Configuration

(a) Omen desktop ubuntu config



(b) Asus ubuntu config

jupyter lab for development. When finished convert to App

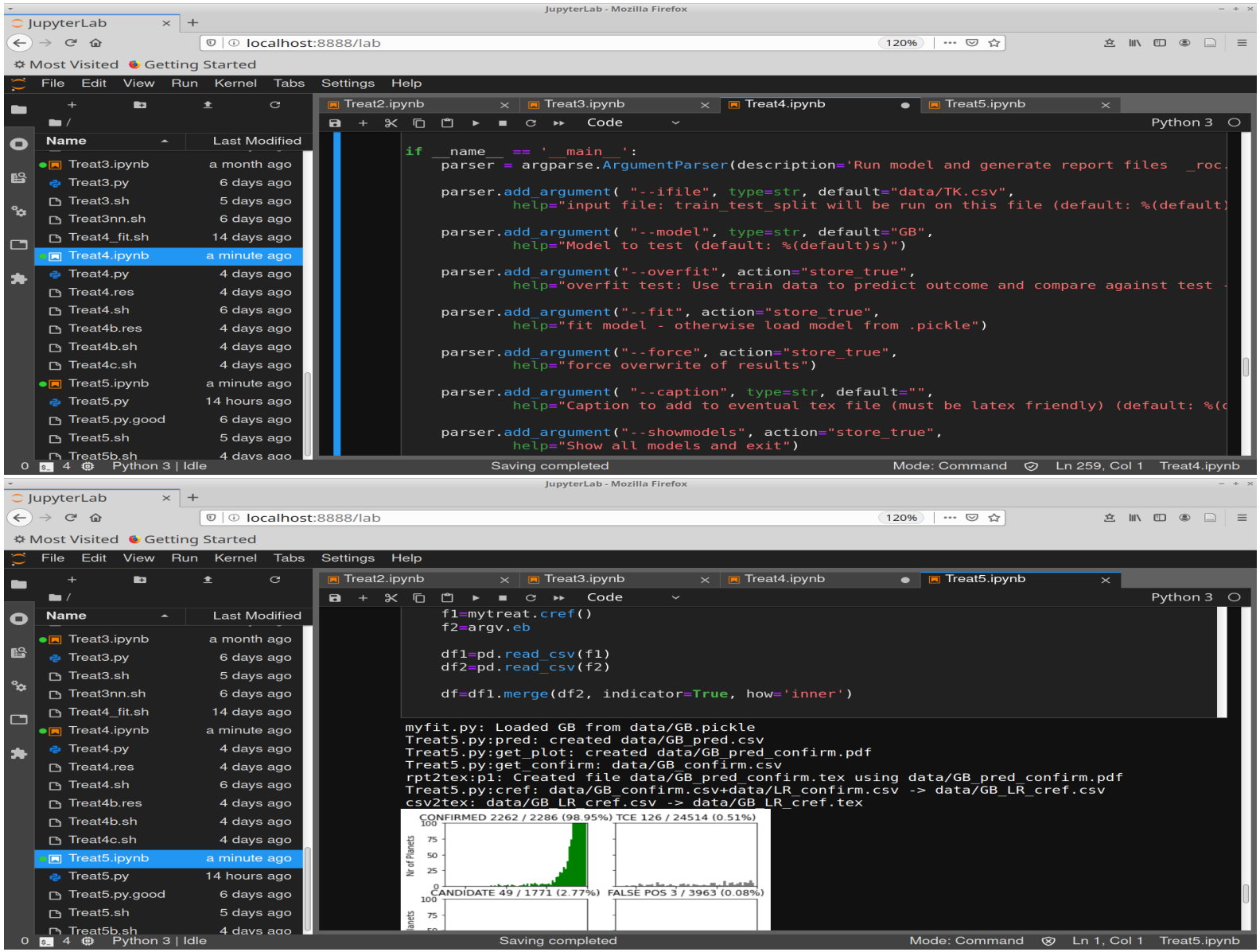


Figure: jupyter lab used for python development

SEMMA pipeline overview

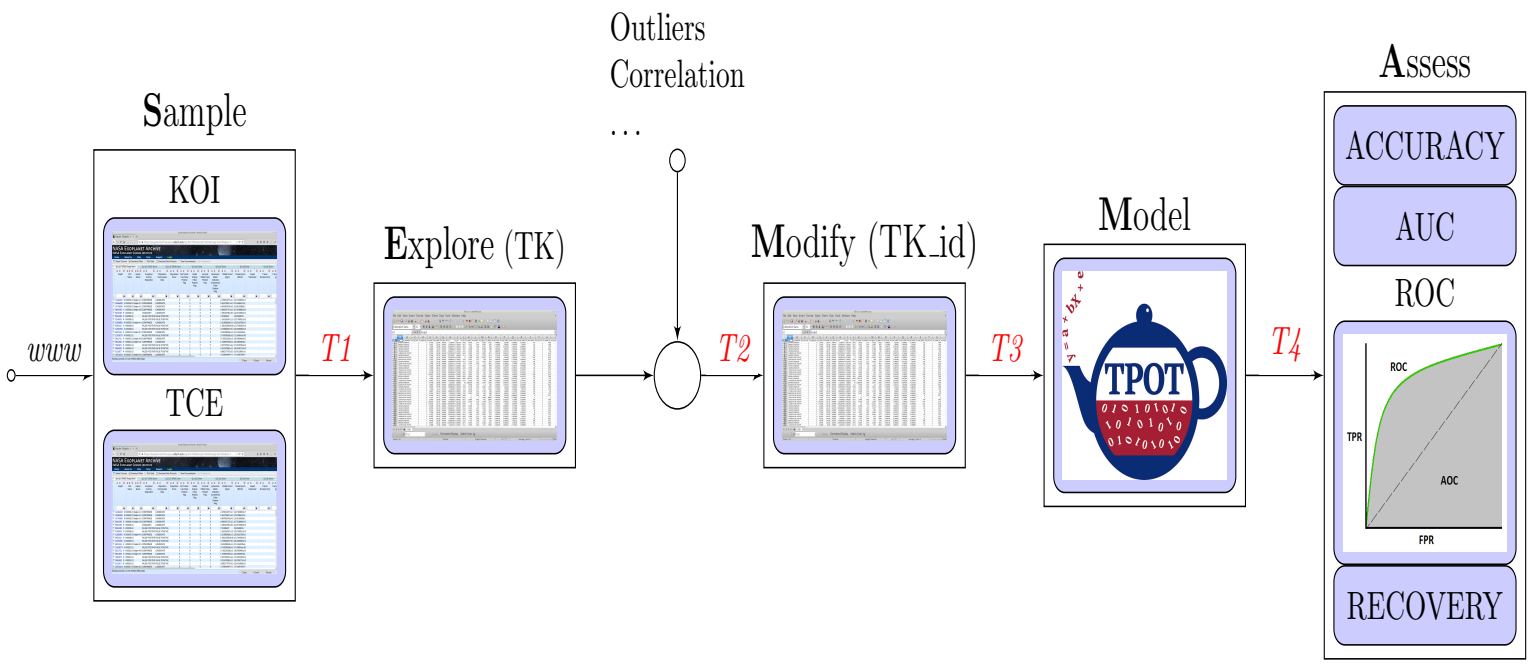
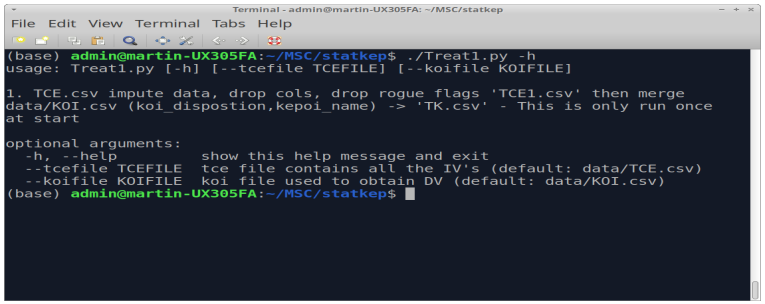
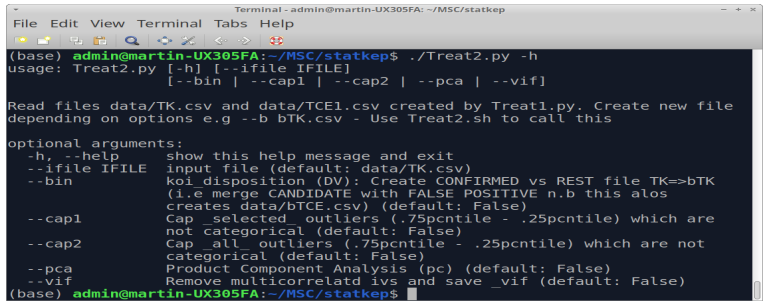


Figure: Treatment pipeline Overview: Each *T*reatment has a defined input and output which is usually a csv file

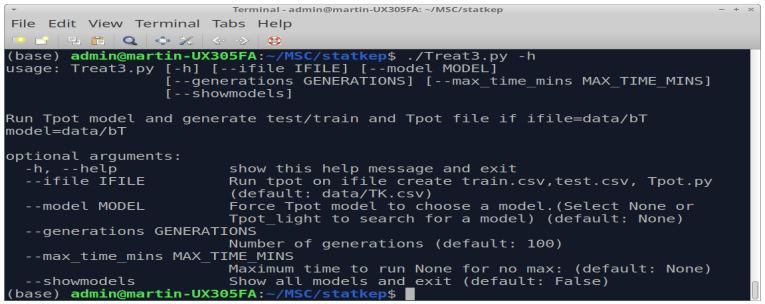
Treat 1,2,3,4,5



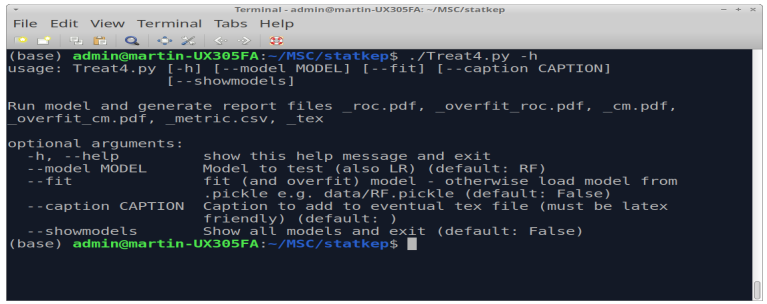
(a) Treat1.py



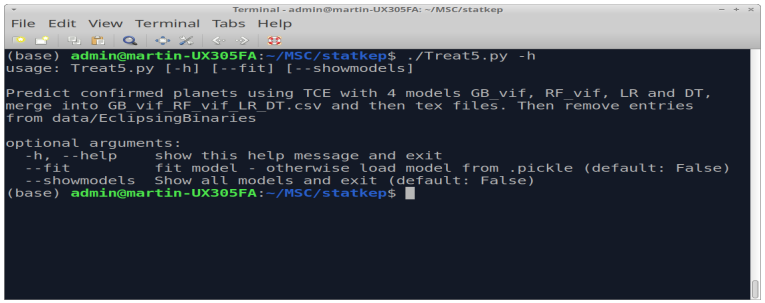
(b) Treat2.py



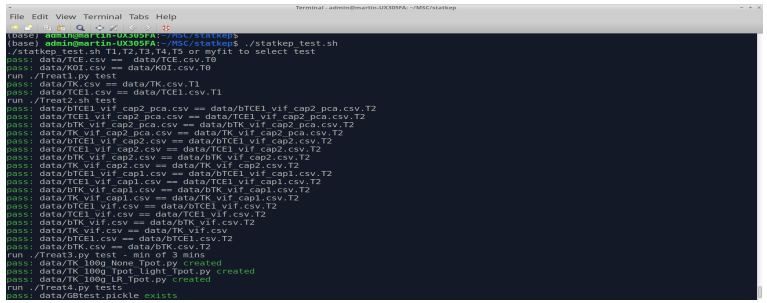
(c) Treat3.py



(d) Treat4.py



(e) Treat5.py



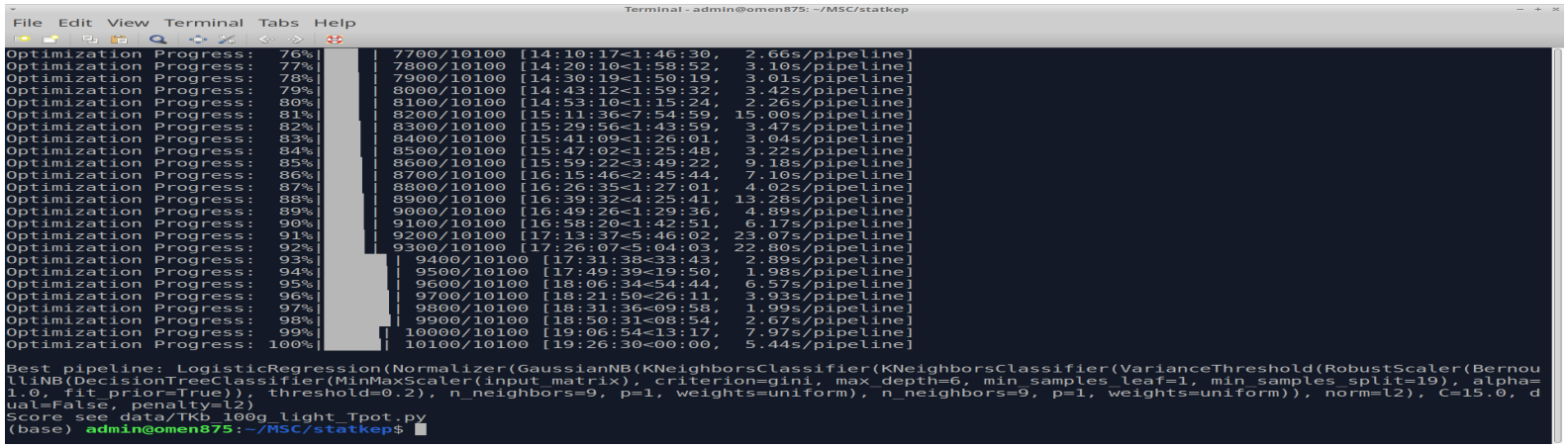
(f) Regresssion Tests

Figure: Treatments and their help files

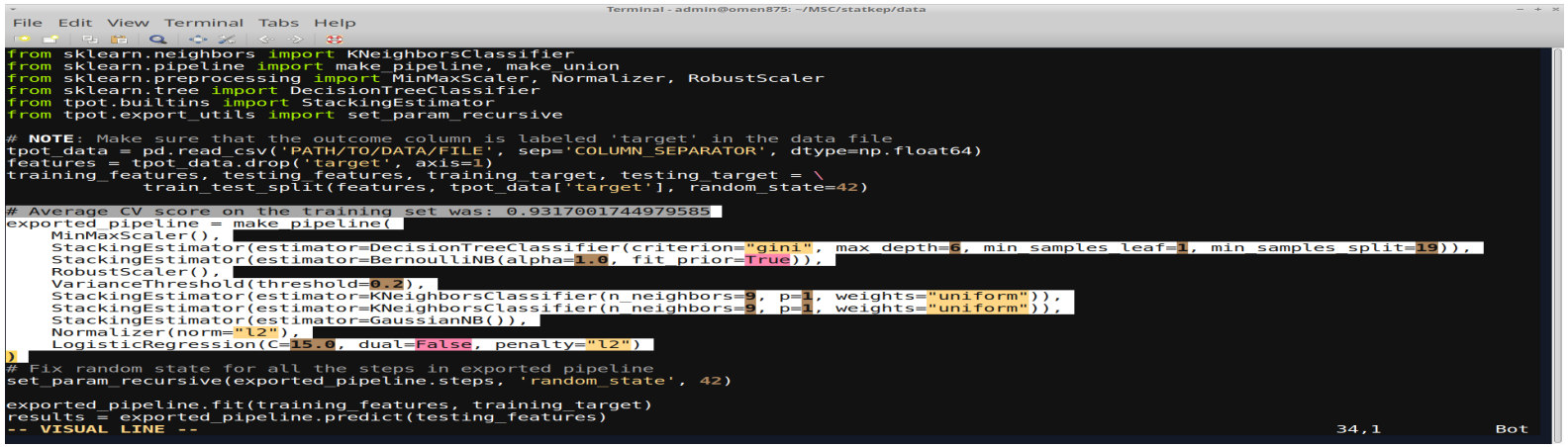
NASA TCE and KOI data

- Transit Crossing Event Data `https://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTbls&config=tce`
- Kepler Object of interest (KOI) Data: `https://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTbls&config=koi`

Treat3.py - Generate code using TPOT



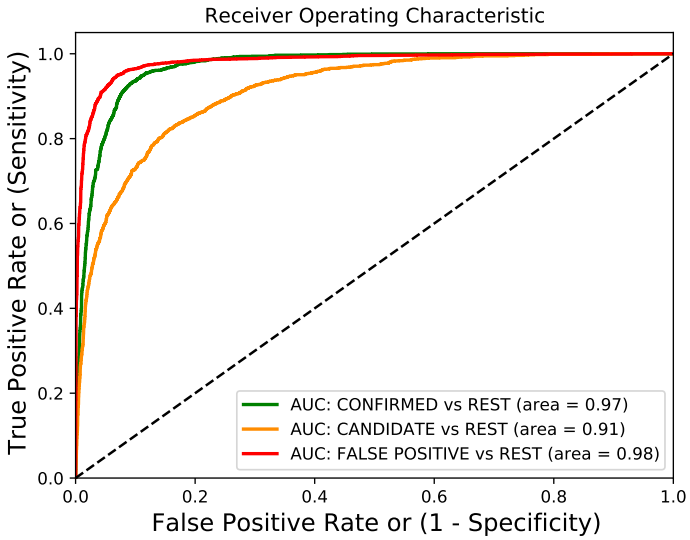
(a) Optimization script duration 19 hours and 26 Minutes

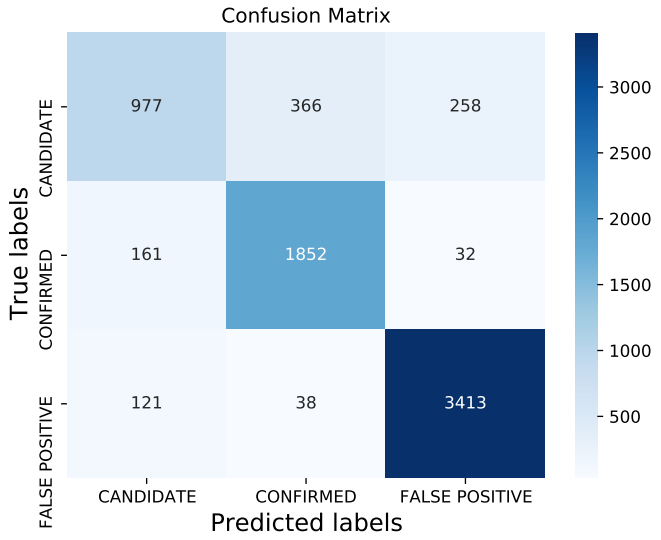


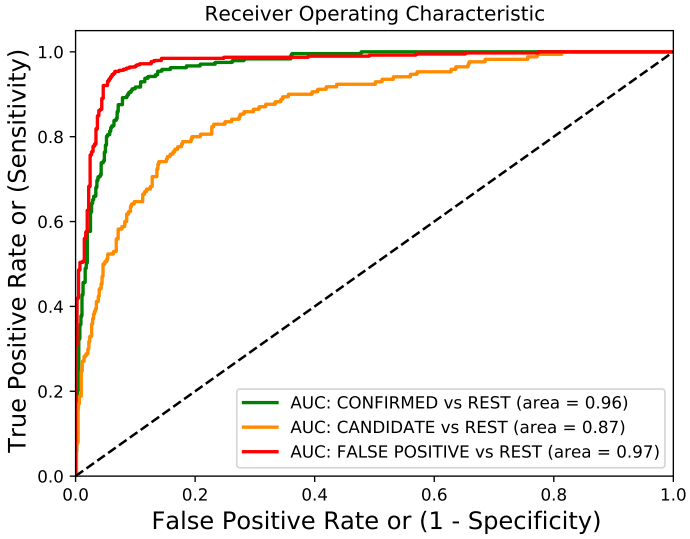
(b) Optimal Model script automatically generated by tpot

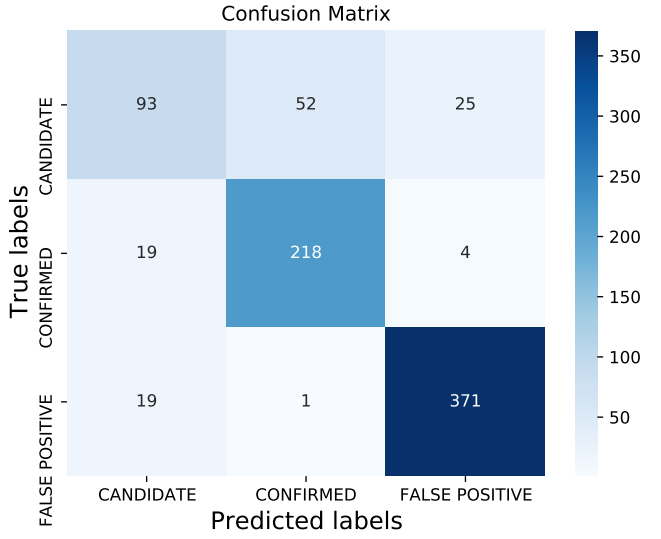
Figure: (a)TPOT generates the optimal script for modelling. screen put long running scripts in background (b) TPOT ouput. In this case Logistic Regression chosen

Treat4.py output –model LR









Treat5.py output –model LR

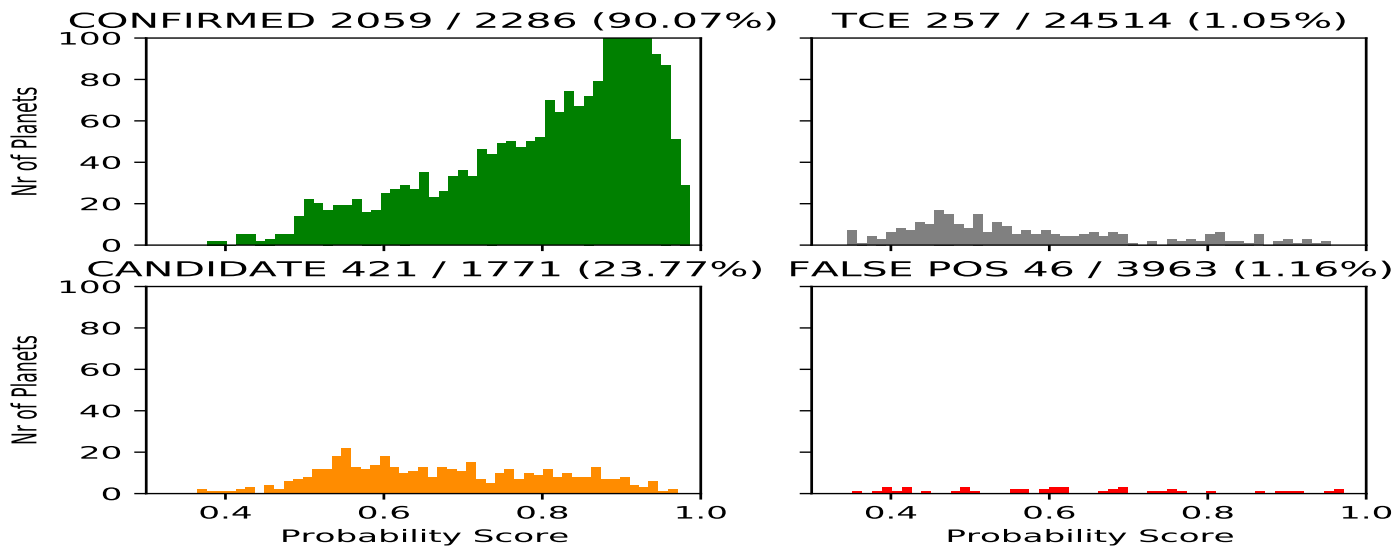


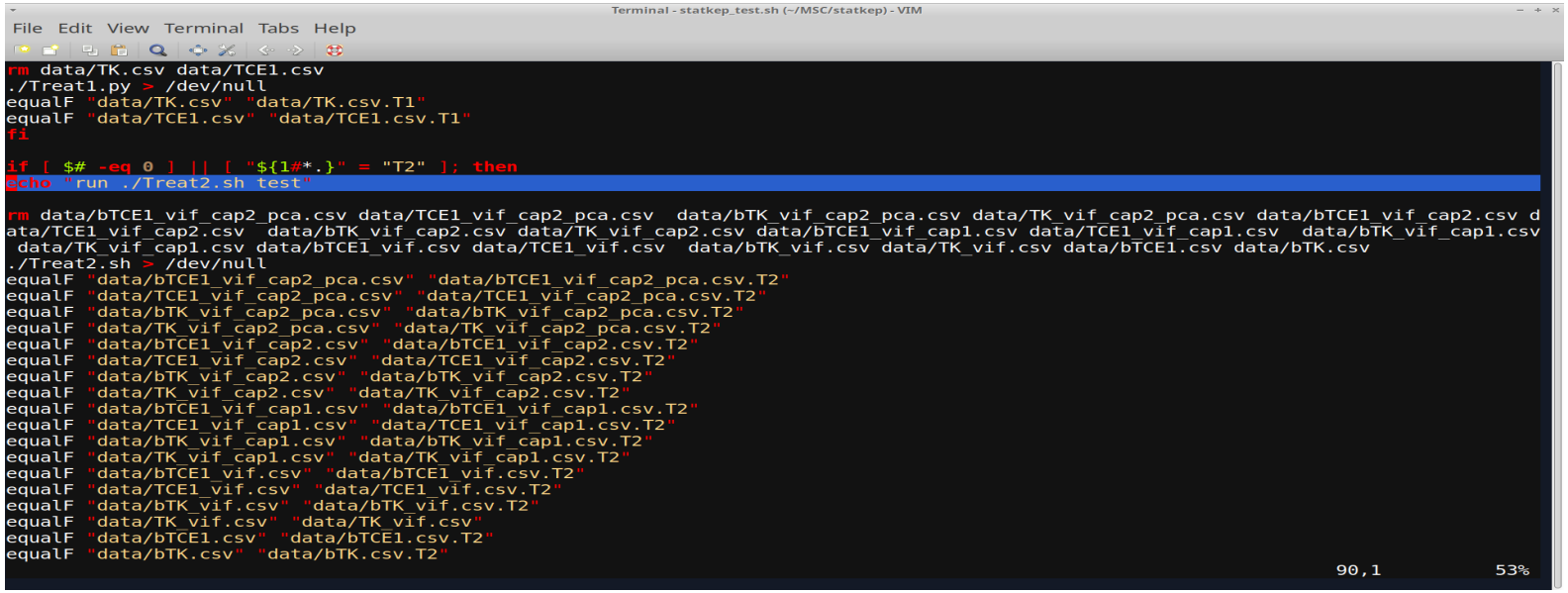
Figure: LR: Planets predicted as CONFIRMED

Result Table

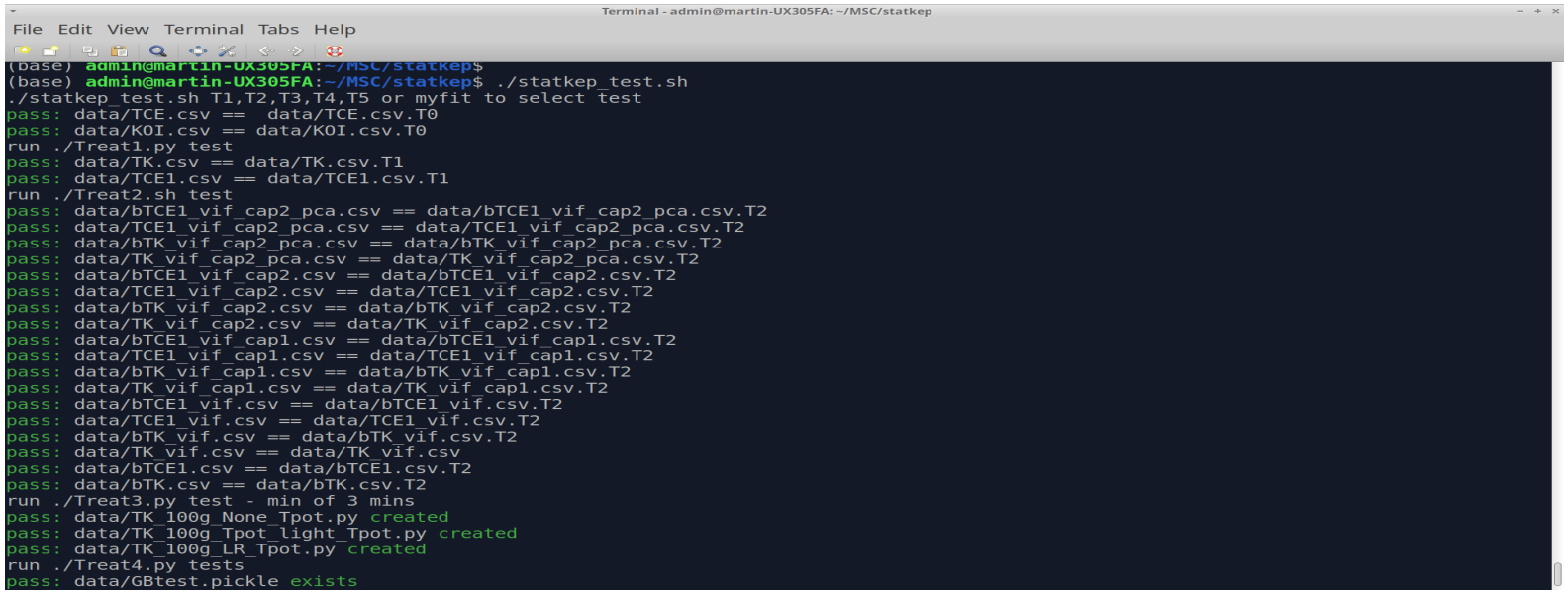
Table: Probability of being a confirmed planet sorted by Logistic Regression

kepid	**plnt num**	**kname**	**dispos**	**GB vif**	**RF vif**	**LR**	**DT**
8456679	2	K00102.02	FP	-	-	0.972	0.951
8480285	1	K00691.01	CAND	-	0.503	0.971	0.948
8804455	2	K02159.02	FP	-	-	0.963	0.986
10788461	1	K03925.01	CAND	-	-	0.959	0.571
8644365	1	K03384.01	CAND	-	-	0.948	0.943
8804845	1	K02039.01	CAND	-	-	0.945	0.919
3831053	1	K00388.01	CAND	-	0.642	0.944	0.966
12505076	1	K02154.01	CAND	-	-	0.944	0.971
4149450	1	K01864.01	CAND	-	-	0.943	0.992
2581316	2	**K03681.02**	CAND	1.0	0.952	0.942	0.932
5709725	2	K00555.02	CAND	-	0.516	0.937	0.919
5374854	2	K00645.02	CAND	-	-	0.93	0.94

Unit Tests using shell scripts



(a) Test script bash source code



All Treatments are object orientated and documented using python __doc__

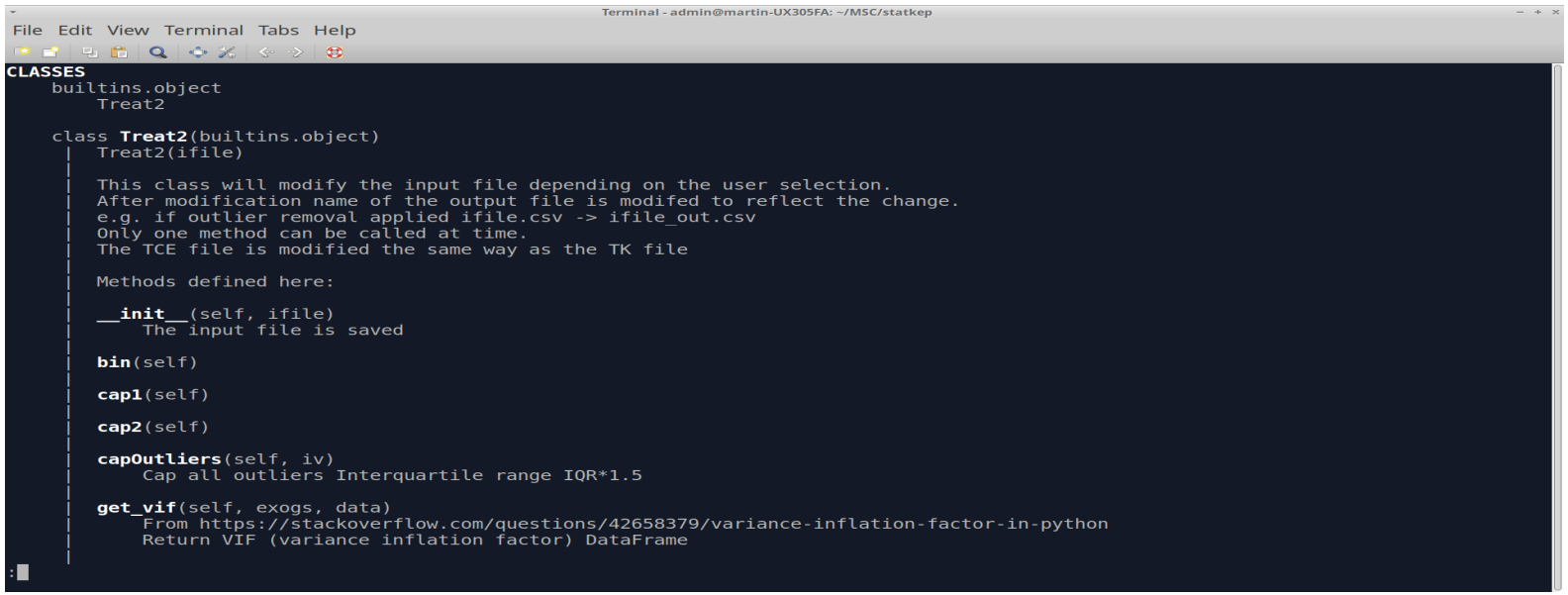


Figure: Treat2.py methods: To view: import Treat2;help(Treat2)

The End