

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

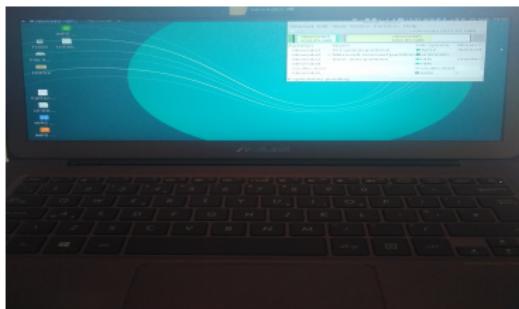
Overview

- 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

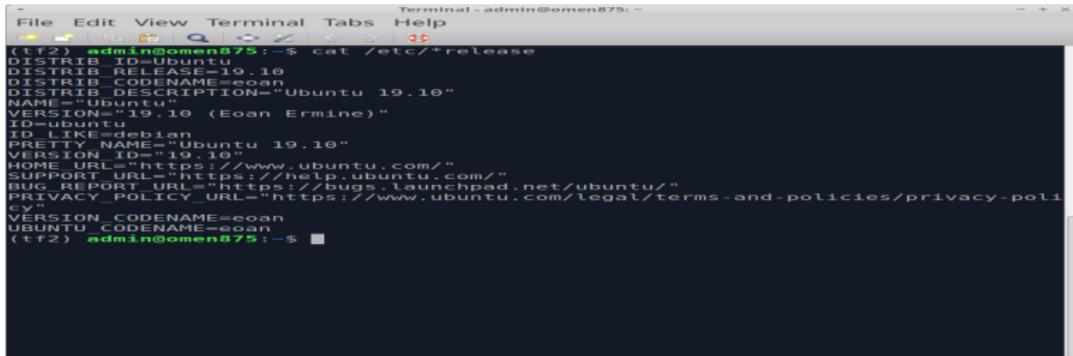


(a) HP - OMEN 875-0051 Gaming PC
2 TB HDD / 256 GB SSD disk
Intel® Core™ i7, RTX 2060



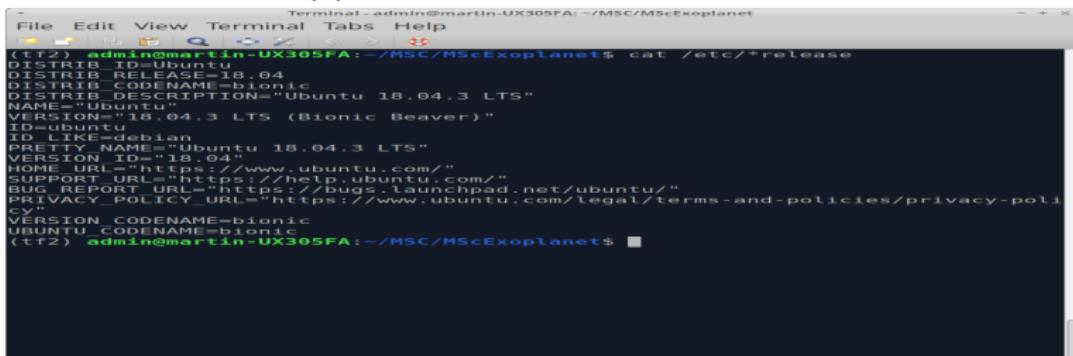
(b) Asus Zenbook UX30FA laptop
1TB SSD disk
Intel® 5Y10 CPU @ 0.80GHz

Ubuntu Configuration



```
File Edit View Terminal Tabs Help
(ttf2) admin@omen875:~ cat /etc/*release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=19.10
DISTRIB_CODENAME=eoan
DISTRIB_DESCRIPTION="Ubuntu 19.10"
NAME="Ubuntu"
VERSION="19.10 (Eoan Ermine)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 19.10"
VERSION_ID="19.10"
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
VERSION_CODENAME=eoan
UBUNTU_CODENAME=eoan
(ttf2) admin@omen875:~
```

(a) Omen desktop ubuntu config



```
File Edit View Terminal Tabs Help
(ttf2) admin@martin-UX305FA:~/MSC/MScExoplanet$ cat /etc/*release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=18.04
DISTRIB_CODENAME=bionic
DISTRIB_DESCRIPTION="Ubuntu 18.04.3 LTS"
NAME="Ubuntu"
VERSION="18.04.3 LTS (Bionic Beaver)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 18.04.3 LTS"
VERSION_ID="18.04"
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
VERSION_CODENAME=bionic
UBUNTU_CODENAME=bionic
(ttf2) admin@martin-UX305FA:~/MSC/MScExoplanet$
```

(b) Asus ubuntu config

jupyter lab for development. When finished convert to App

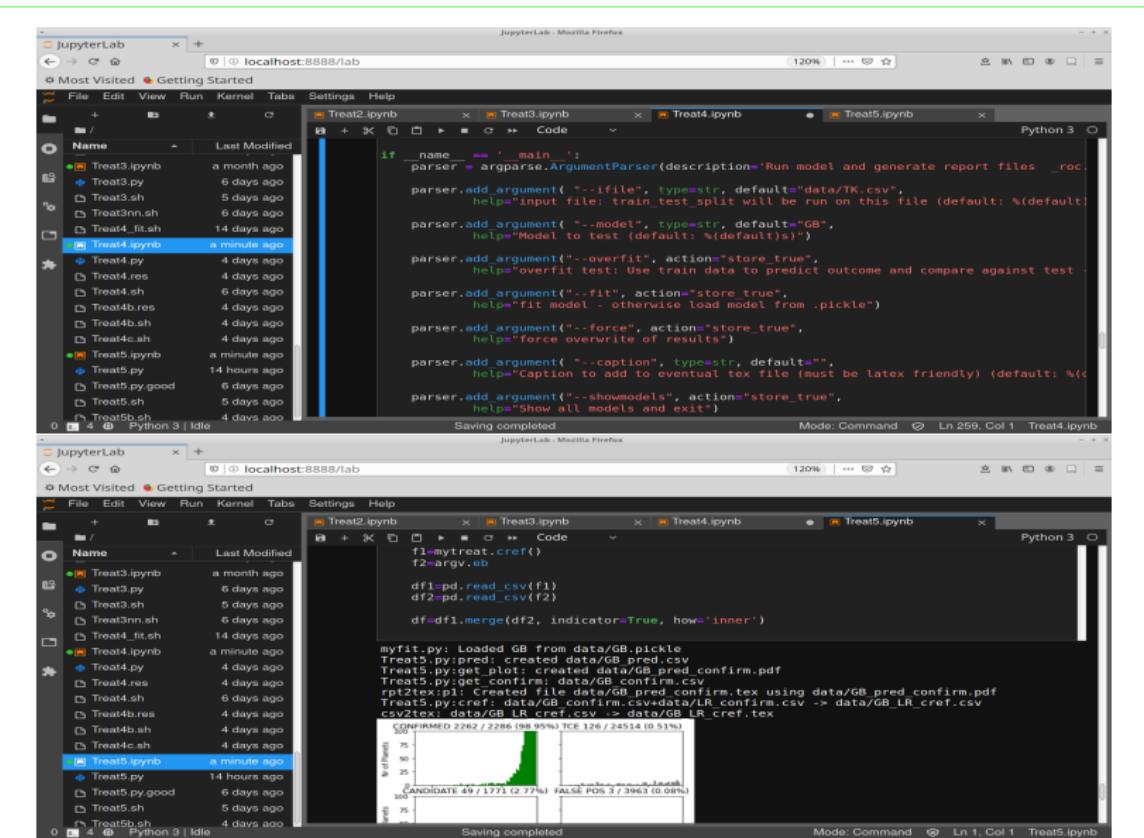


Figure: jupyter lab used for python development

SEMMA pipeline overview

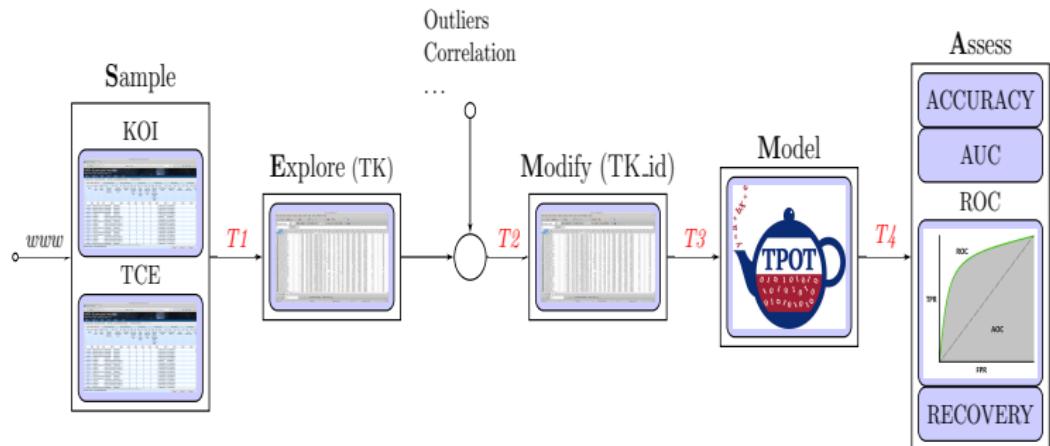


Figure: Treatment pipeline Overview: Each **Treatment** has a defined input and output which is usually a csv file

Treat 1,2,3,4,5

```
Terminal address: /dev/martin-UX305FA:/~MSC/statkeps
```

```
File Edit View Terminal Tabs Help
(base) admiringmartin-UX305FA:~/MSC/statkeps$ ./Treat1.py -h
usage: Treat1.py [-h] [--tcofile TCOFILE] [--koifile KOIFILE]
                  [TCE.csv input data, drop cols, drop rogue flags 'TCE1.csv' then merge
                   data/KOI.csv (kol_disposition,Kepol_name) --> TK.csv]  This is only run once
                  or start
optional arguments:
  -h, --help            show this help message and exit
  --tcofile TCOFILE    tco file used to obtain DV (default: data/TCE.csv)
  --koifile KOIFILE    kol file used to obtain DV (default: data/KOI.csv)
(base) admiringmartin-UX305FA:~/MSC/statkeps$
```

(a) Treat1.py

```
Terminal address: /dev/martin-UX305FA:/~MSC/statkeps
```

```
File Edit View Terminal Tabs Help
(base) admiringmartin-UX305FA:~/MSC/statkeps$ ./Treat3.py -h
usage: Treat3.py [-h] [--ifile IFILE] [--model MODEL]
                  [--generations GENERATIONS] [--max_time_mins MAX_TIME_MINS]
                  [--showmodels]
Run Tpot model and generate test/train and Tpot file if ifile=data/bT
Model=data/bT
optional arguments:
  -h, --help            show this help message and exit
  --ifile IFILE         Run Tpot to generate train.csv, Tpot.py
                        (default: data/TK.csv)
  --model MODEL         Force Tpot model to choose a model (Select None or
                        Tpot_light to search for a model) (default: None)
  --generations GENERATIONS
                        Number of generations (default: 100)
  --max_time_mins MAX_TIME_MINS
                        Maximum time to run None for no max: (default: None)
  --showmodels          Show all models and exit (default: False)
(base) admiringmartin-UX305FA:~/MSC/statkeps$
```

(c) Treat3.py

```
Terminal address: /dev/martin-UX305FA:/~MSC/statkeps
```

```
File Edit View Terminal Tabs Help
(base) admiringmartin-UX305FA:~/MSC/statkeps$ ./Treat5.py -h
usage: Treat5.py [-h] [--models MODELS] [--showmodels]
                  [data/EclipsingBinaries.csv --> EclipsingBinaries]
predict confirmed planets using TCE with 4 models GB vif, RF vif, LR and DT,
merge into GB vif RF vif LR DT.csv and then Tex files. Then remove entries
from data/EclipsingBinaries
optional arguments:
  -h, --help            show this help message and exit
  --models MODELS       FIT model otherwise load model from .pickle (default: False)
  --showmodels          Show all models and exit (default: False)
(base) admiringmartin-UX305FA:~/MSC/statkeps$
```

(e) Treat5.py

```
Terminal address: /dev/martin-UX305FA:/~MSC/statkeps
```

```
File Edit View Terminal Tabs Help
(base) admiringmartin-UX305FA:~/MSC/statkeps$ ./Treat2.py -h
usage: Treat2.py [-h] [--tcofile TCOFILE] [--koifile KOIFILE]
                  [data/TK.csv and data/TCE1.csv created by Treat1.py. Create new file
                   depending on options e.g. -b BTK.csv - Use Treat2.sh to call this
                  ]
optional arguments:
  -h, --help            show this help message and exit
  --tcofile TCOFILE    tco file used to obtain DV (default: data/TCE.csv)
  --koifile KOIFILE    kol disposition (DV): Create CONFIRMED vs REST file TK=>bTK
                      (i.e Merge CANDIDATE with FALSE CONFIRMED n.b this also
                      creates CONFIRMED file if candidate=False)
  --cap1               Cap selected outliers (.75pcntile - .25pcntile) which are
                      categorical (default: True)
  --cap2               Cap all outliers (.75pcntile - .25pcntile) which are not
                      categorical (default: False)
  --pca                Remove multicorrelated ivs and save _vif (default: False)
  --vif                Remove multicorrelated ivs and save _vif (default: False)
(base) admiringmartin-UX305FA:~/MSC/statkeps$
```

(b) Treat2.py

```
Terminal address: /dev/martin-UX305FA:/~MSC/statkeps
```

```
File Edit View Terminal Tabs Help
(base) admiringmartin-UX305FA:~/MSC/statkeps$ ./Treat4.py -h
usage: Treat4.py [-h] [--model MODEL] [--fit] [--caption CAPTION]
                  [--showmodels]
Run model and generate report files _roc.pdf, _overfit_roc.pdf, _cm.pdf,
_overfit_cm.pdf, _metric.csv, _tex
optional arguments:
  -h, --help            Show this help message and exit
  --model MODEL         Model to test (also LR) (default: RF)
  --fit                Fit model to data (default: False)
  --caption CAPTION    Add caption to eventual tex file (must be latex
                        friendly)
  --showmodels          Show all models and exit (default: False)
(base) admiringmartin-UX305FA:~/MSC/statkeps$
```

(d) Treat4.py

```
Terminal address: /dev/martin-UX305FA:/~MSC/statkeps
```

```
File Edit View Terminal Tabs Help
(base) admiringmartin-UX305FA:~/MSC/statkeps$ ./Treat5.py -h
usage: Treat5.py [-h] [--models MODELS] [--showmodels]
                  [data/EclipsingBinaries.csv --> EclipsingBinaries]
predict confirmed planets using TCE with 4 models GB vif, RF vif, LR and DT,
merge into GB vif RF vif LR DT.csv and then Tex files. Then remove entries
from data/EclipsingBinaries
optional arguments:
  -h, --help            show this help message and exit
  --models MODELS       FIT model otherwise load model from .pickle (default: False)
  --showmodels          Show all models and exit (default: False)
(base) admiringmartin-UX305FA:~/MSC/statkeps$
```

(f) Regression 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

```
File Edit View Terminal Tabs Help
25
Optimization Progress: 76% 7798/10100 [14: 10]:/c=3.66_59, 2.66%/pipeline]
Optimization Progress: 77% 7799/10100 [14: 20]:/c=1.58_52, 2.10%/pipeline]
Optimization Progress: 78% 7900/10100 [14: 30]:/c=1.50_19, 3.01%/pipeline]
Optimization Progress: 79% 8000/10100 [14: 40]:/c=1.44_53, 3.90%/pipeline]
Optimization Progress: 80% 8100/10100 [14: 53]:/c=1.15_24, 2.26%/pipeline]
Optimization Progress: 81% 8200/10100 [15: 11]:/c=7.54_59, 15.09%/pipeline]
Optimization Progress: 82% 8300/10100 [15: 21]:/c=6.89_19, 15.15%/pipeline]
Optimization Progress: 83% 8400/10100 [15: 41]:/c=0.99_126_01, 3.04%/pipeline]
Optimization Progress: 84% 8500/10100 [15: 51]:/c=2.52_131, 3.54%/pipeline]
Optimization Progress: 85% 8600/10100 [15: 59]:/c=22.3_49_22, 9.18%/pipeline]
Optimization Progress: 86% 8700/10100 [16: 15]:/c=4.62_245_44, 7.10%/pipeline]
Optimization Progress: 87% 8800/10100 [16: 35]:/c=0.88_17_10, 2.21%/pipeline]
Optimization Progress: 88% 8900/10100 [16: 39]:/c=32_44_25_41, 13.28%/pipeline]
Optimization Progress: 89% 9000/10100 [16: 50]:/c=1.65_70_49_01, 2.64%/pipeline]
Optimization Progress: 90% 9100/10100 [16: 58]:/c=0.29_1.42_51, 6.17%/pipeline]
Optimization Progress: 91% 9200/10100 [17: 13]:/c=7.5_46_62, 23.07%/pipeline]
Optimization Progress: 92% 9300/10100 [17: 23]:/c=0.54_46_62, 2.53%/pipeline]
Optimization Progress: 93% 9400/10100 [17: 31]:/c=38_53_43, 2.89%/pipeline]
Optimization Progress: 94% 9500/10100 [17: 35]:/c=21.3_38_62, 2.83%/pipeline]
Optimization Progress: 95% 9600/10100 [18: 06]:/c=3.64_54_44, 6.57%/pipeline]
Optimization Progress: 96% 9700/10100 [18: 21]:/c=5_26_11, 3.93%/pipeline]
Optimization Progress: 97% 9800/10100 [18: 30]:/c=0.33_10_19, 1.69%/pipeline]
Optimization Progress: 98% 9900/10100 [18: 50]:/c=33<0>_54, 2.67%/pipeline]
Optimization Progress: 99% 10000/10100 [19: 00]:/c=20.5_16_49, 10.37%/pipeline]
Optimization Progress: 100% 10100/10100 [19: 20]:/c=10_20_30<0>, 5.44%/pipeline]

Grit pipeline: LogisticRegression(Normalizer(GaussianNB(KNeighborsClassifier(VarianceThreshold(RobustScaler(BernoulliNB(DecisionTreeClassifier(MinMaxScaler(input_matrix), criterion='gini', max_depth=6, min_samples_leaf=1, min_samples_split=10), alpha=1.0, prior=True), threshold=0.2), n_neighbors=9, p=1, weights=uniform), n_neighbors=9, p=1, weights=uniform), norm=l2), C=15.0, dual=False, tol=1e-05, max_iter=200)))
Score see data/TKD_100g_light.Tpot.py
(base) admin@monolith:~/MLC/statkeys$
```

(a) Optimization script duration 19 hours and 26 Minutes

```
File Edit View Terminal Tabs Help
25
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import make_pipeline, make_union
from sklearn.preprocessing import MinMaxScaler, Normalizer, RobustScaler
from tpot.builtins import DecisionTreeClassifier
from tpot.builtins import StackingEstimator
from tpot.export_utils import set_param_recursive

# NOTE: Make sure that the outcome column is labeled 'target' in the data file
# when loading data from scikit-learn. If this isn't the case, please add the
# label as follows:
#
# features = tpot.data.drop(['target'], axis=1)
# training_features = features.append(tpot.data['target'], axis=1)
# training_target = \
#     training_features.pop('target', axis=0)
# train, test = train_test_split(features, tpot.data['target'], random_state=42)

# Average CV score on the training set was: 0.9517001744979585
exported_pipeline = make_pipeline(
    MinMaxScaler(),
    StackingEstimator(estimator=DecisionTreeClassifier(criterion="gini", max_depth=9, min_samples_leaf=1, min_samples_split=13)),
    VarianceThreshold(threshold=0.2),
    StackingEstimator(estimator=BernoulliNB(alpha=3.0, fit_prior=True)),
    RobustScaler(),
    StackingEstimator(estimator=KNeighborsClassifier(n_neighbors=4, p=1, weights="uniform")),
    StackingEstimator(estimator=KNeighborsClassifier(n_neighbors=4, p=1, weights="uniform")),
    StackingEstimator(estimator=GaussianNB()),
    NullSelection(),
    LogisticRegression(C=15.0, dual=False, penalty="l2"))

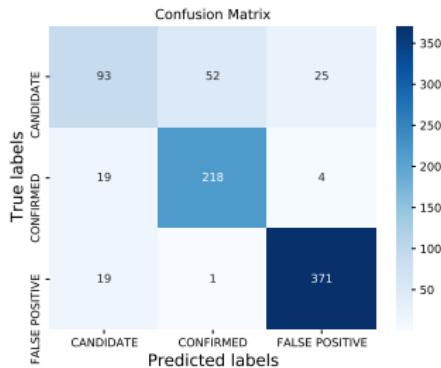
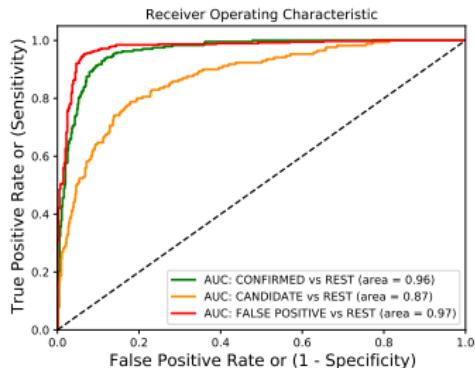
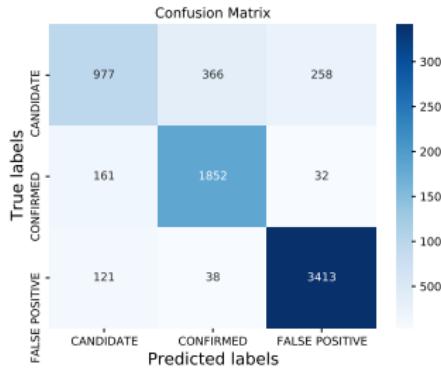
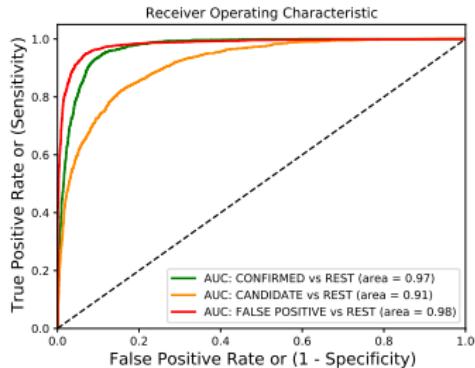
# Fix random state for all the steps in exported pipeline
set_params_recursive(exported_pipeline.steps, 'random_state', 42)

exported_pipeline.fit(training_features, training_target)
res = tpot.export_pipeline.predict(testing_features)
# VISUAL_LIME --
```

(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

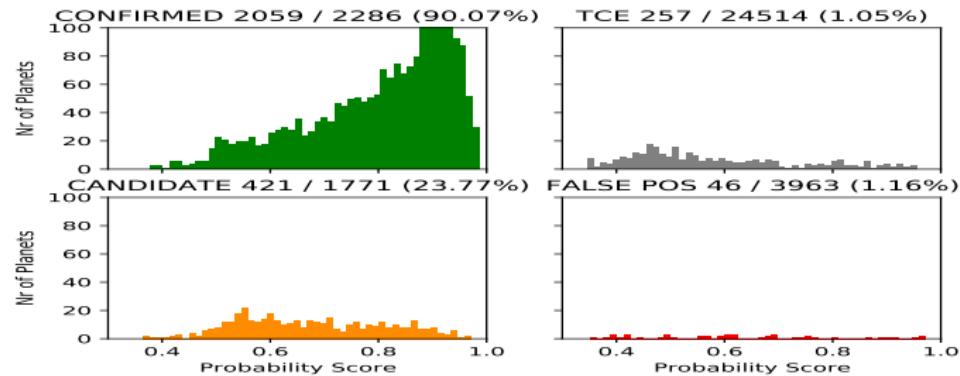


Figure: LR: Planets predicted as CONFIRMED

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

```
File Edit View Terminal Tabs Help
[base] [file] [new] [open] [close] [refresh]
# data/TK.csv data/TCE1.csv
./Treat1.py > /dev/null
equalF "data/TK.csv" "data/TK.csv.T1"
equalF "data/TCE1.csv" "data/TCE1.csv.T1"
fi
if [ $# -eq 0 ] || [ "$1" == "T2" ]; then
echo "run ./Treat2.sh test"

rm data/bTCE1_vif_cap2_pca.csv data/TCE1_vif_cap2_pca.csv data/bTk_vif_cap2_pca.csv data/TK_vif_cap2_pca.csv data/bTCE1_vif_cap2.csv
rm data/TK_vif_cap2_pca.csv data/bTCE1_vif_cap1.csv data/TCE1_vif_cap1.csv data/bTk_vif_cap1.csv
./Treat2.sh > /dev/null

equalF "data/bTCE1_vif_cap2_pca.csv" "data/bTCE1_vif_cap2_pca.csv.T2"
equalF "data/TCE1_vif_cap2_pca.csv" "data/TCE1_vif_cap2_pca.csv.T2"
equalF "data/TK_vif_cap2_pca.csv" "data/TK_vif_cap2_pca.csv.T2"
equalF "data/bTk_vif_cap2_pca.csv" "data/bTk_vif_cap2_pca.csv.T2"
equalF "data/bTCE1_vif_cap2_pca.csv" "data/bTCE1_vif_cap2_pca.csv.T2"
equalF "data/TCE1_vif_cap2_pca.csv" "data/TCE1_vif_cap2_pca.csv.T2"
equalF "data/TK_vif_cap2_pca.csv" "data/TK_vif_cap2_pca.csv.T2"
equalF "data/bTk_vif_cap2_pca.csv" "data/bTk_vif_cap2_pca.csv.T2"
equalF "data/bTCE1_vif_cap1.csv" "data/TCE1_vif_cap1.csv.T2"
equalF "data/TCE1_vif_cap1.csv" "data/TCE1_vif_cap1.csv.T2"
equalF "data/bTk_vif_cap1.csv" "data/bTk_vif_cap1.csv.T2"
equalF "data/bTCE1_vif_cap1.csv" "data/bTCE1_vif_cap1.csv.T2"
equalF "data/bTk_vif_cap1.csv" "data/bTk_vif_cap1.csv.T2"
equalF "data/TK_vif_cap1.csv" "data/TK_vif_cap1.csv.T2"
equalF "data/bTCE1_vif.csv" "data/bTCE1_vif.csv.T2"
equalF "data/TCE1_vif.csv" "data/TCE1_vif.csv.T2"
equalF "data/bTk_vif.csv" "data/bTk_vif.csv.T2"
equalF "data/TK_vif.csv" "data/TK_vif.csv.T2"
equalF "data/bTCE1.csv" "data/bTCE1.csv.T2"
equalF "data/bTk.csv" "data/bTk.csv.T2"

Terminal - statkep_test.sh (~/MSC/statkep) - VIM
90,1 53%
```

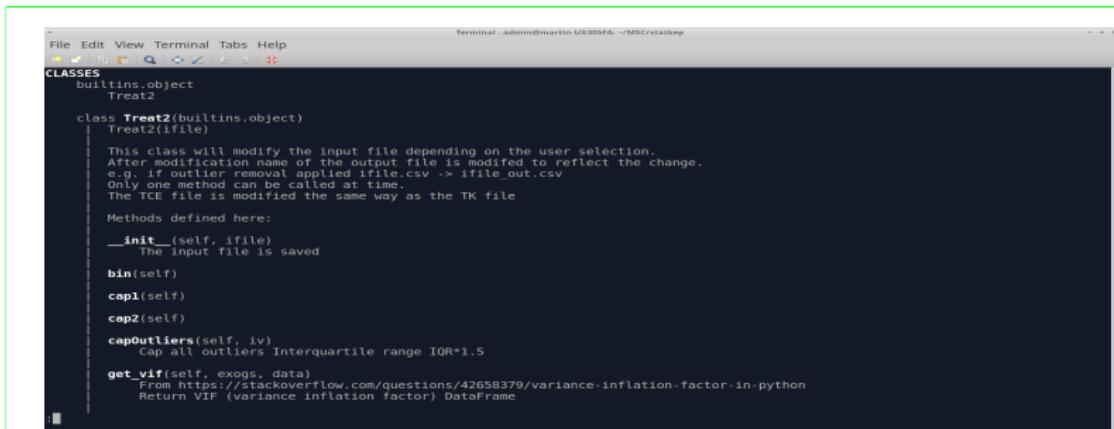
(a) Test script bash source code

```
File Edit View Terminal Tabs Help
[base] [file] [new] [open] [close] [refresh]
(base) [admin@martin-UX305FA:~/MSC/statkep]
(base) admin@martin-UX305FA:~/MSC/statkep$ ./statkep test.sh
./statkep test.sh T1,T2,T3,T4,T5 or myfif to select test
pass: data/TCE.csv == data/TCE.csv.T0
pass: data/KO1.csv == data/KO1.csv.T0
run ./Treat1.py test
pass: data/TK.csv == data/TK.csv.T1
pass: data/TCE1.csv == data/TCE1.csv.T1
run ./Treat2.sh test
pass: data/bTCE1_vif_cap2_pca.csv == data/bTCE1_vif_cap2_pca.csv.T2
pass: data/TCE1_vif_cap2_pca.csv == data/TCE1_vif_cap2_pca.csv.T2
pass: data/bTk_vif_cap2_pca.csv == data/bTk_vif_cap2_pca.csv.T2
pass: data/TK_vif_cap2_pca.csv == data/TK_vif_cap2_pca.csv.T2
pass: data/bTCE1_vif_cap2_pca.csv == data/bTCE1_vif_cap2_pca.csv.T2
pass: data/TCE1_vif_cap2_pca.csv == data/TCE1_vif_cap2_pca.csv.T2
pass: data/bTk_vif_cap2_pca.csv == data/bTk_vif_cap2_pca.csv.T2
pass: data/TK_vif_cap2_pca.csv == data/TK_vif_cap2_pca.csv.T2
pass: data/bTCE1_vif_cap1.csv == data/bTCE1_vif_cap1.csv.T2
pass: data/TCE1_vif_cap1.csv == data/TCE1_vif_cap1.csv.T2
pass: data/bTk_vif_cap1.csv == data/bTk_vif_cap1.csv.T2
pass: data/TK_vif_cap1.csv == data/TK_vif_cap1.csv.T2
pass: data/bTCE1_vif.csv == data/bTCE1_vif.csv.T2
pass: data/TCE1_vif.csv == data/TCE1_vif.csv.T2
pass: data/bTk_vif.csv == data/bTk_vif.csv.T2
pass: data/TK_vif.csv == data/TK_vif.csv.T2
pass: data/bTCE1.csv == data/bTCE1.csv.T2
pass: data/bTk.csv == data/bTk.csv.T2

(base) [admin@martin-UX305FA:~/MSC/statkep]$ ./statkep test.sh T1,T2,T3,T4,T5
./statkep test.sh T1,T2,T3,T4,T5 or myfif to select test
pass: data/TK_100g_NoneTpot.py == created
pass: data/TK_100g_LightTpot.py == created
pass: data/TK_100g_HighTpot.py == created
run ./Treat4.py test
pass: data/gBtest.pickle exists

Terminal - admin@martin-UX305FA: ~/MSC/statkep
90,1 53%
```

All Treatments are object orientated and documented using python __doc__



The screenshot shows a terminal window with a green border. The title bar reads "Terminal - admin@martin:UX309FA: ~/MSCrstakew". The menu bar includes File, Edit, View, Terminal, Tabs, Help, and several icons. The main area displays Python code for the `Treat2` class:

```
File Edit View Terminal Tabs Help
CLASSES
    builtins.object
        Treat2
    class Treat2(builtins.object)
        |   Treat2(ifile)
        |
        |   This class will modify the input file depending on the user selection.
        |   After modification name of the output file is modified to reflect the change.
        |   e.g. if outlier removal applied ifile.csv -> ifile_out.csv
        |   Only one method can be called at time.
        |   The TCE file is modified the same way as the TK file
        |
        | Methods defined here:
        |   __init__(self, ifile)
        |       The input file is saved
        |
        |   bin(self)
        |
        |   cap1(self)
        |
        |   cap2(self)
        |
        |   capoutliers(self, iv)
        |       Cap all outliers Interquartile range IQR*1.5
        |
        |   get_vif(self, exogs, data)
        |       From https://stackoverflow.com/questions/42658379/variance-inflation-factor-in-python
        |       Return VIF (variance inflation factor) DataFrame
```

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

The End