Identifying Kepler Objects

Applied Machine Learning 2021 Final Project

Niels Bohr Institute, University of Copenhagen

17th June 2021

Moritz Bilstein

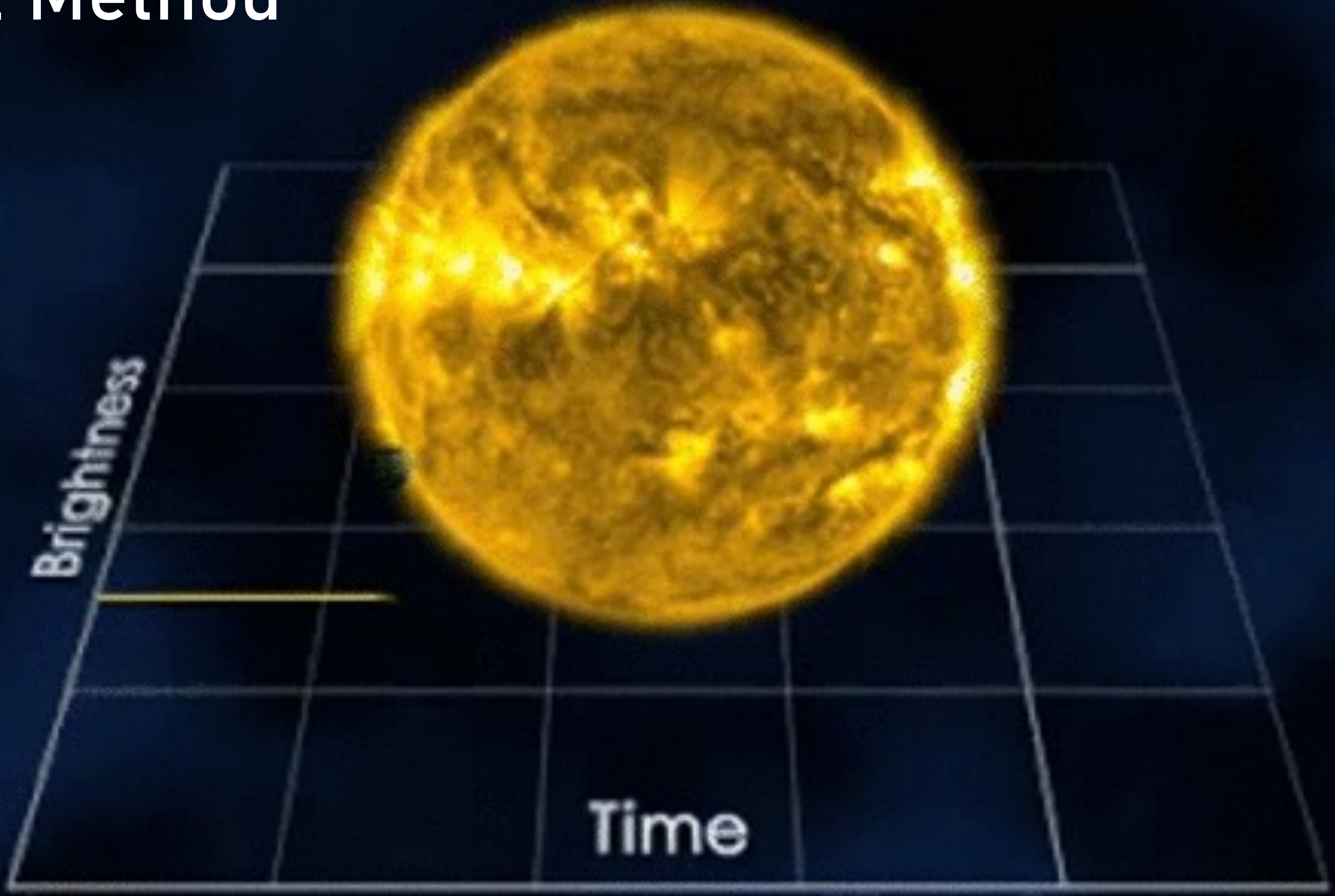
Beatriz Campos Estrada

Carl Gustav Henning Hansen

Marco Merusi

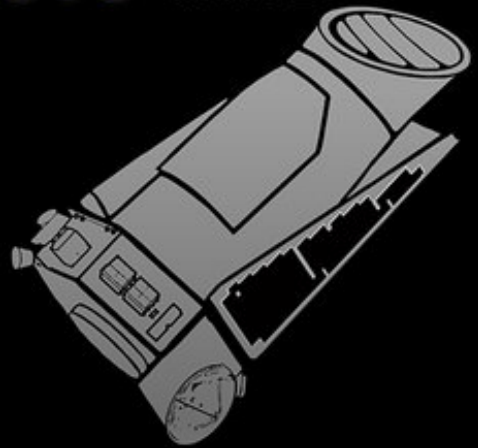
Vittorio Sguazzo

Transit Method



Kepler
BY THE NUMBERS

9.6 YEARS IN SPACE



530,506 STARS OBSERVED

2,662 PLANETS CONFIRMED

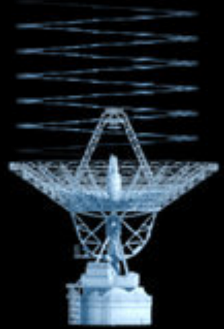
61 SUPERNOVAE DOCUMENTED

FROM EARLIEST STAGES OF EXPLOSION

2 MISSIONS COMPLETED

3.12 GALLONS FUEL USED

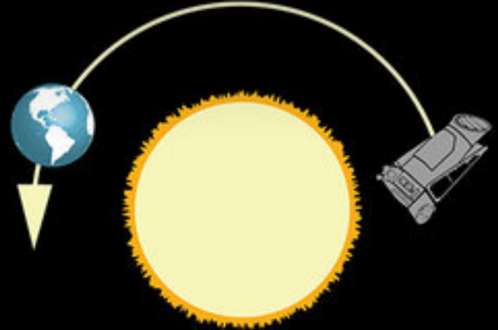
678 GB SCIENCE DATA COLLECTED



2,946 SCIENTIFIC PAPERS PUBLISHED

732,128 COMMANDS EXECUTED

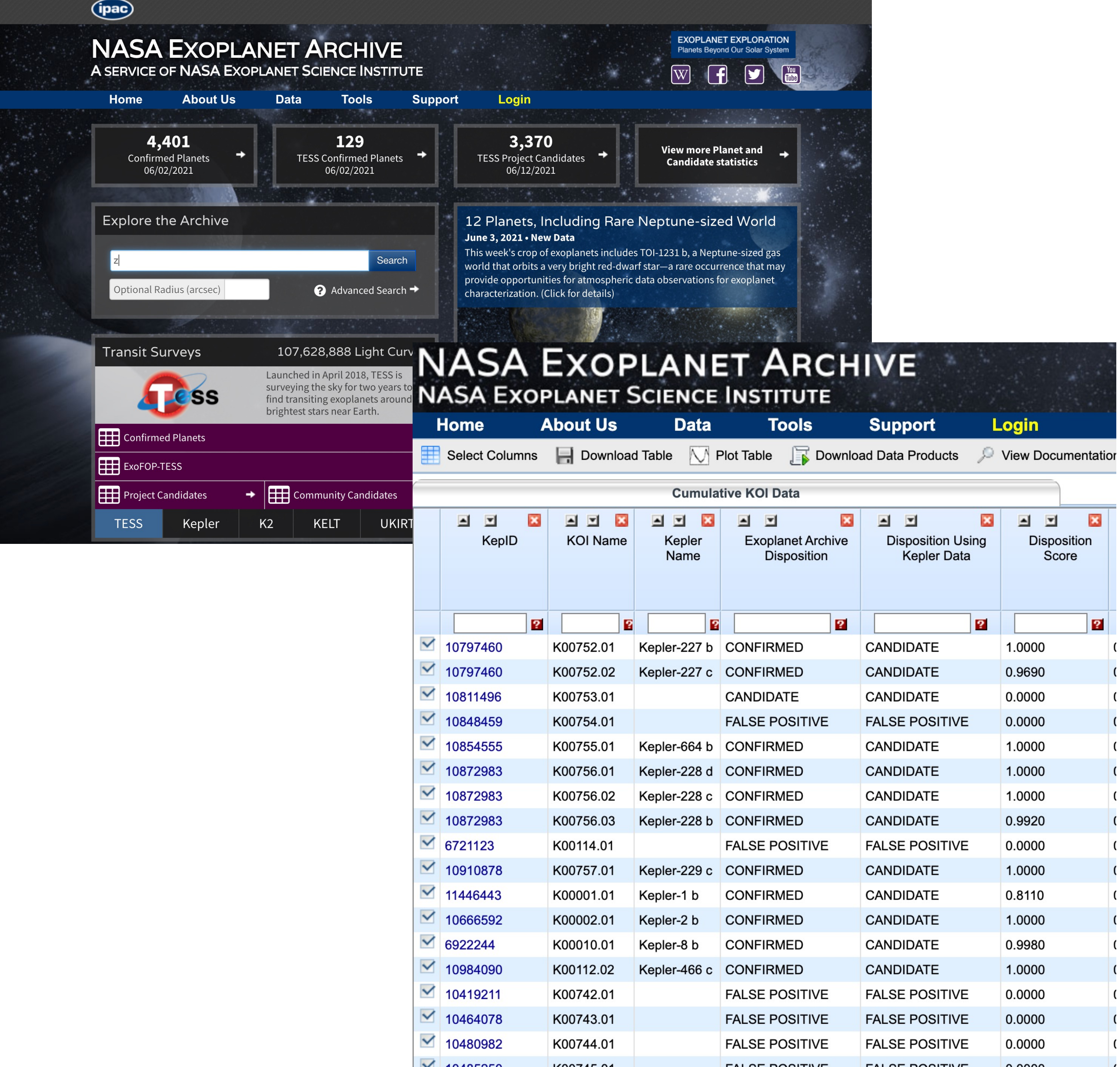
94 MILLION MILES AWAY



www.nasa.gov/kepler

As of October 24, 2018

@NASAKepler

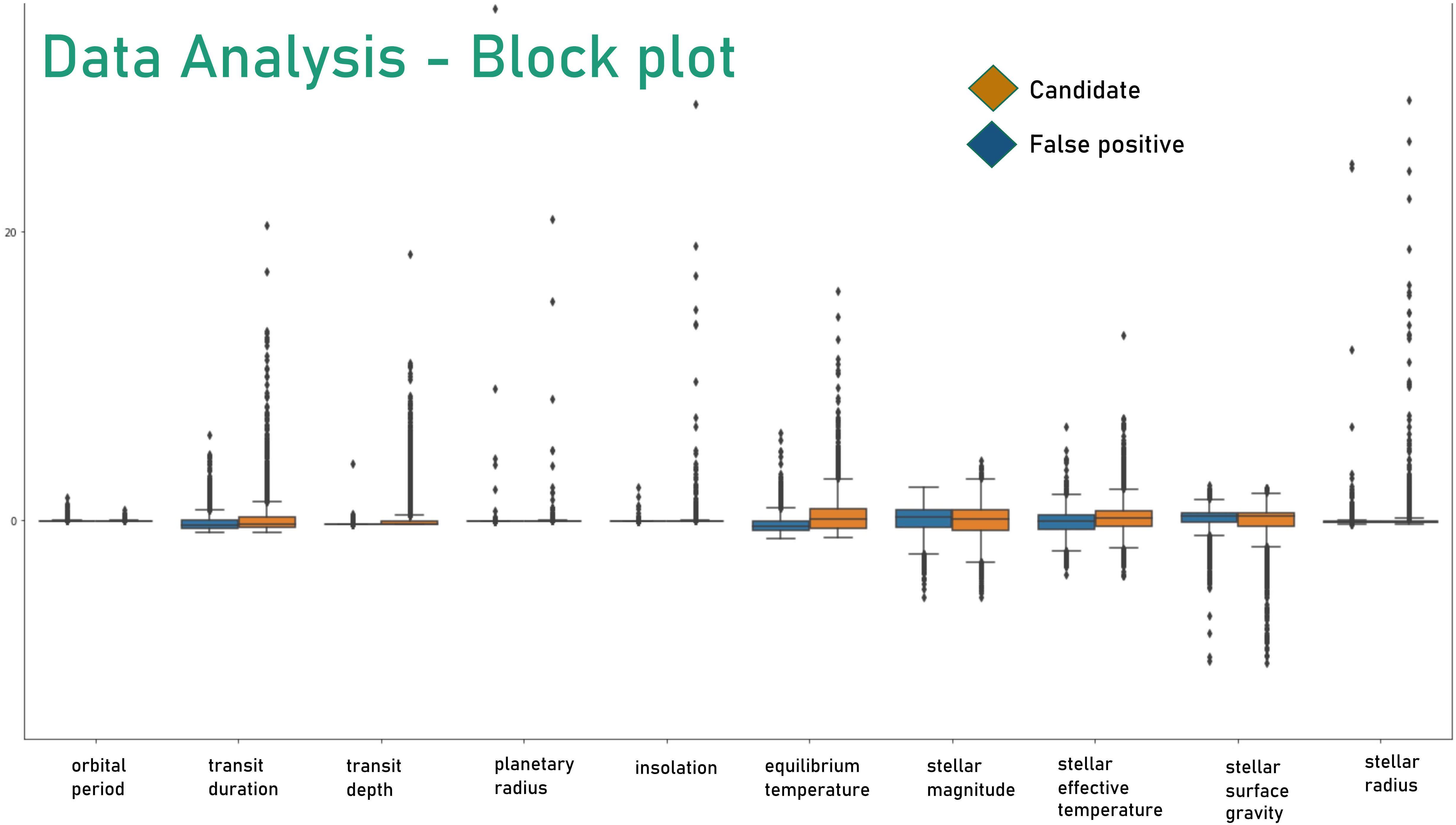


Two problems:

Classification: Planetary Candidates vs False Positives

Regression: Planetary Radii prediction

Data Analysis - Block plot

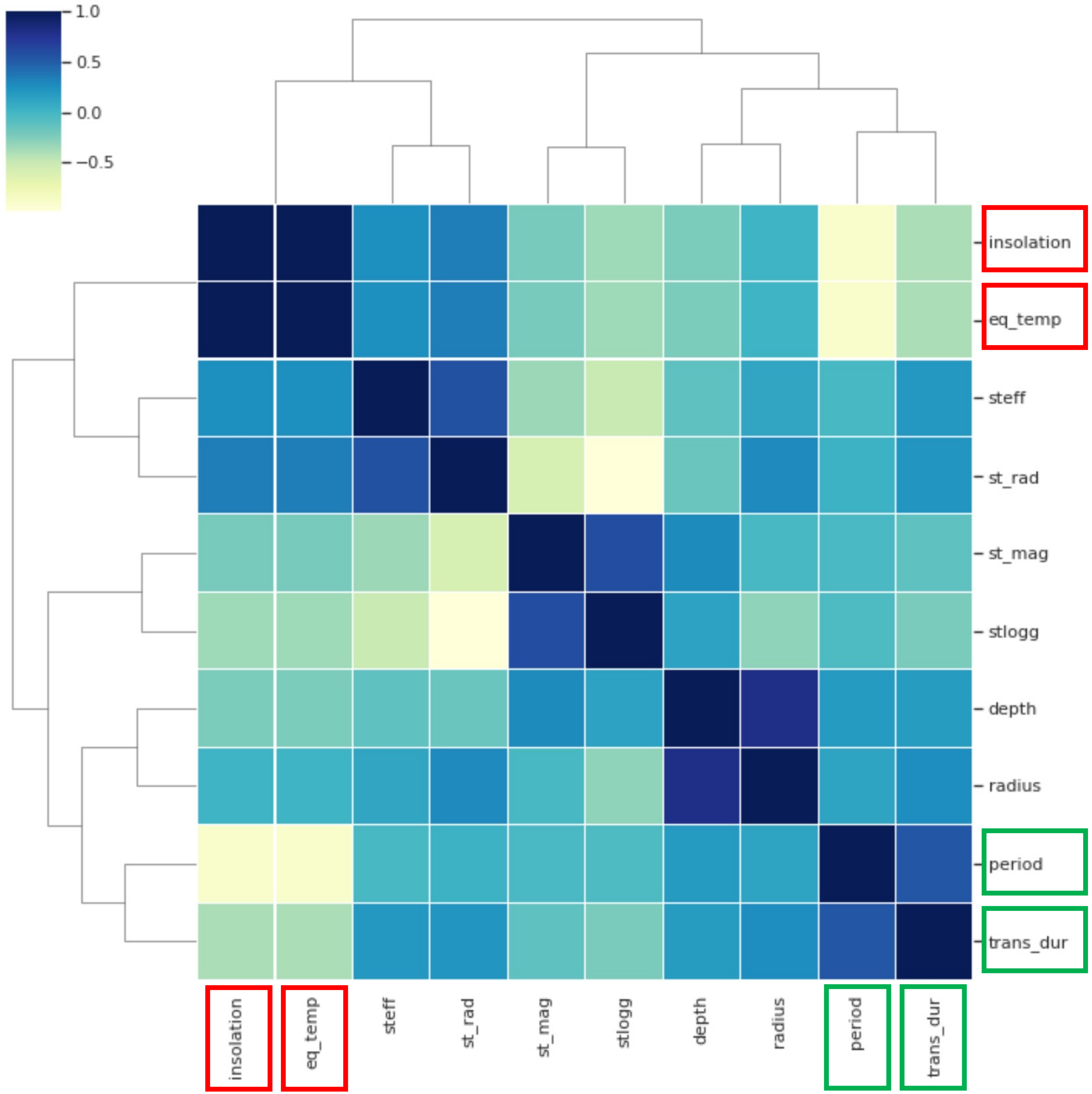


Data Analysis – Correlation Matrix

- Spearman Correlation.

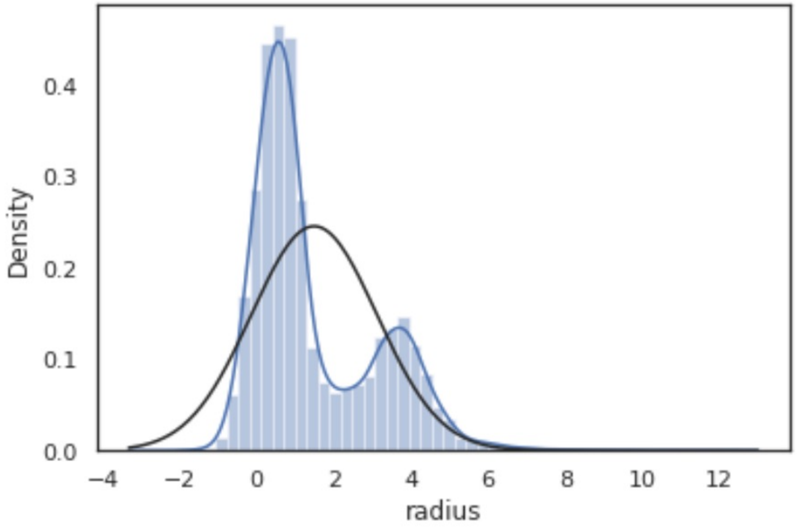
- From two highly correlated features can choose one to leave out:

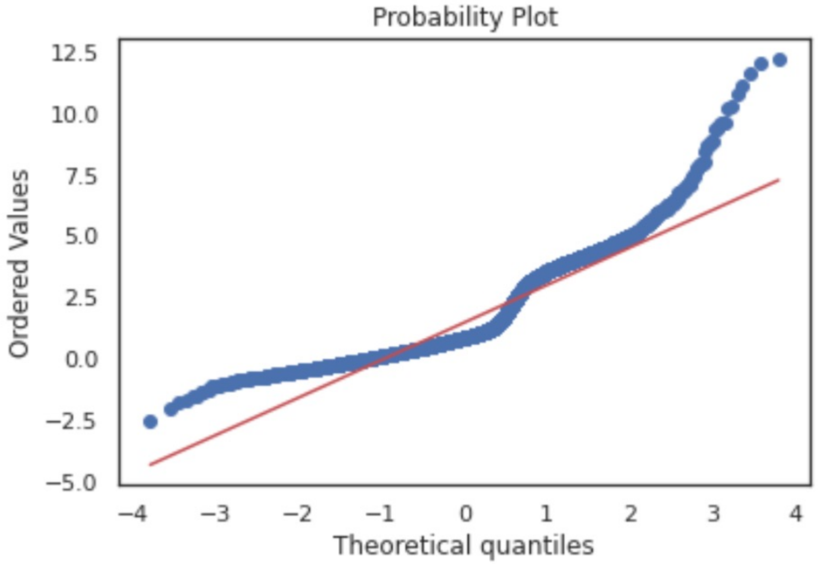
    - E.g.: The insolation is highly correlated to the planet's equilibrium temperature.
    - However, the matrix shows a wrong low correlation between the orbital period and the isolation.



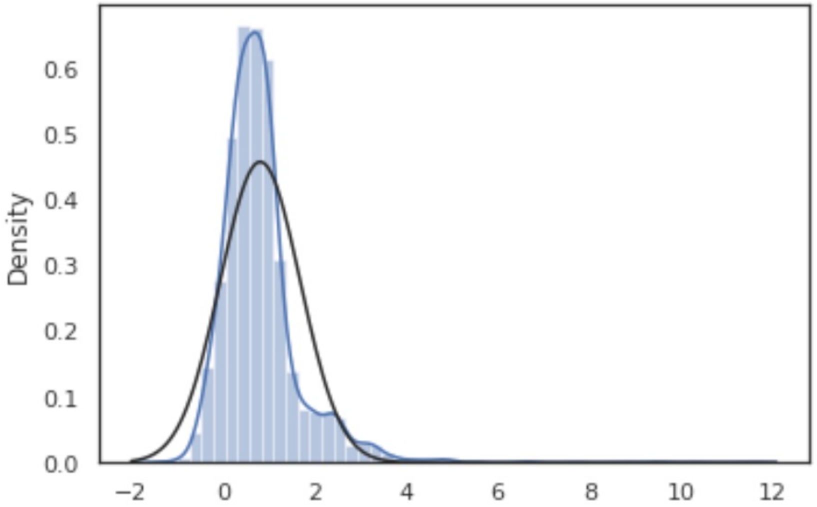
Probability density of planetary radii

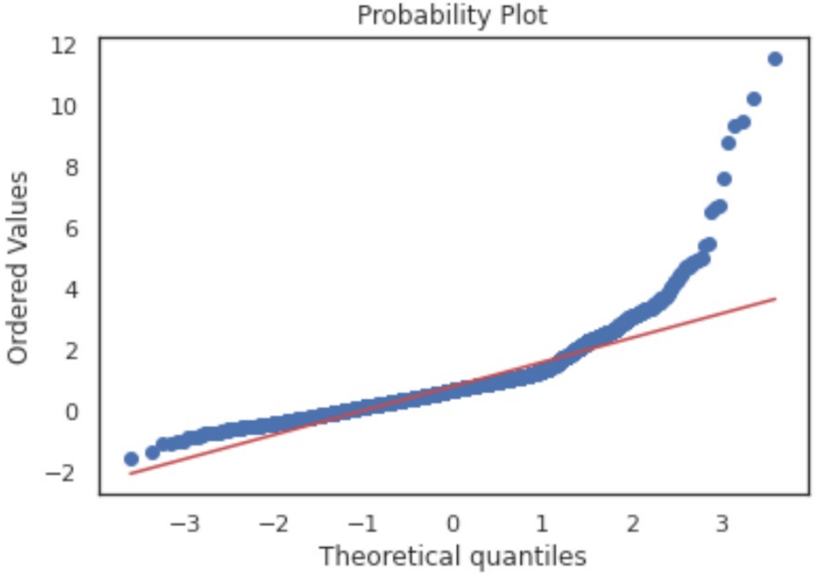
Candidates and false positives





Candidates





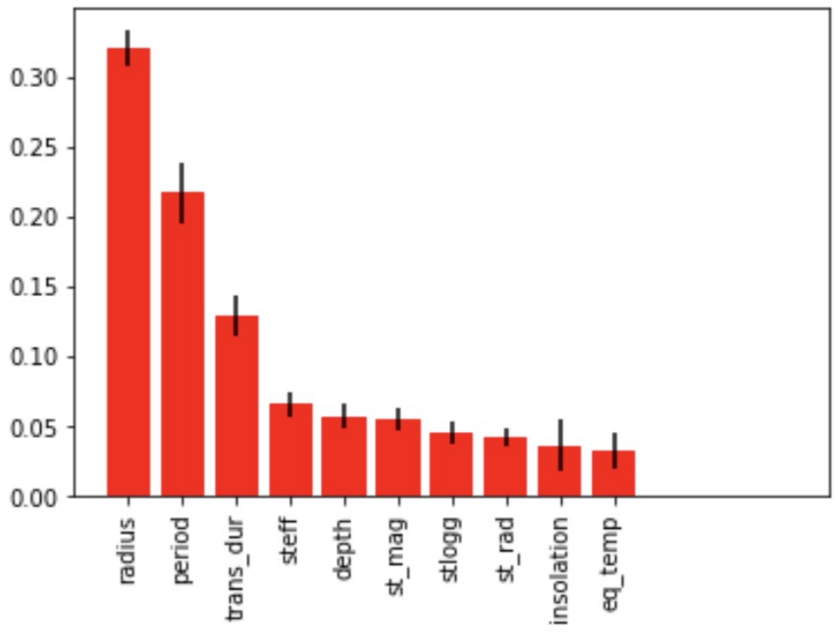
Classification – finding false positives

Random Forest

Main hyperparameters:

- max_depth = 20
- max_features = 10
- n_estimators = 300

Feature importance



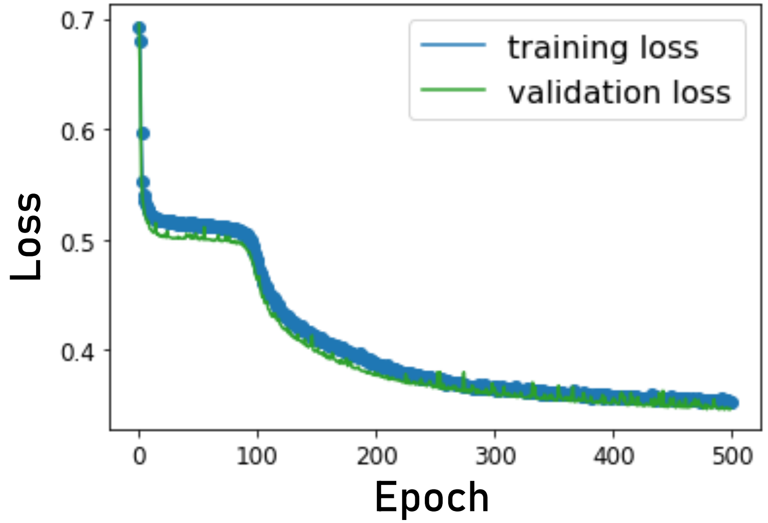
PyTorch Neural Network

Structure of the neural network:

- 1 input layer
- 2 hidden layers, 8 nodes each
- 1 output layer

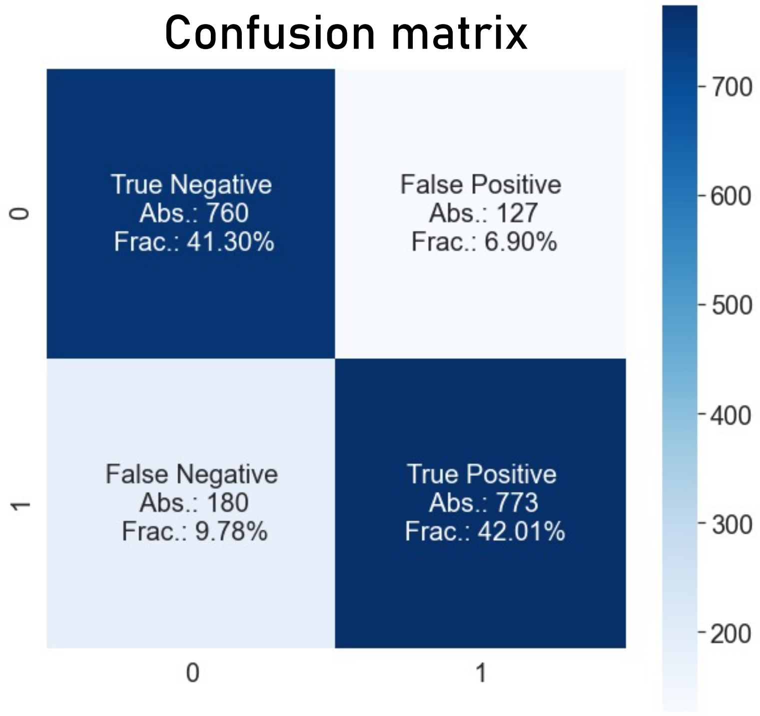
Main hyperparameters:

- learning_rate = 2e-3
- batch_size = 25
- n_epochs = 500



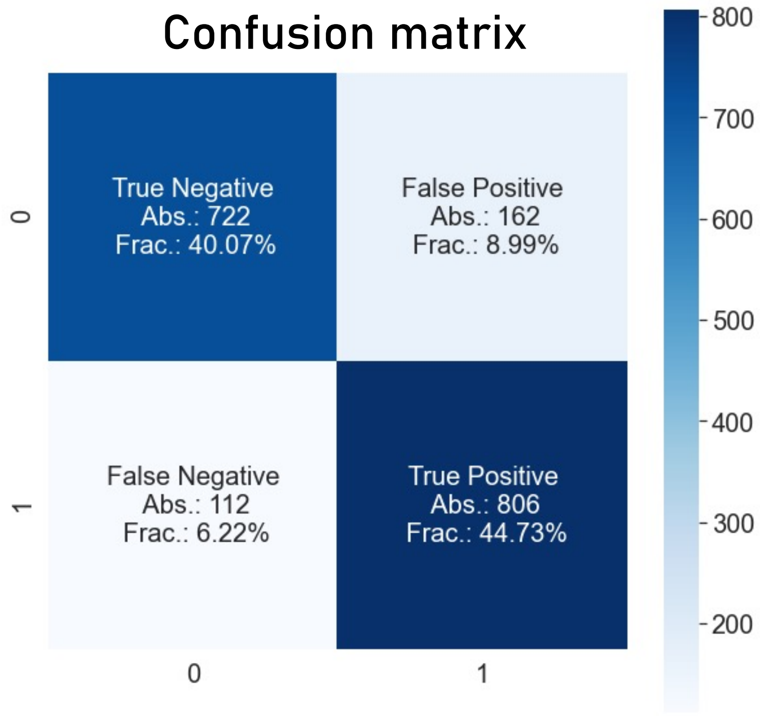
Classification – finding false positives

Random Forest



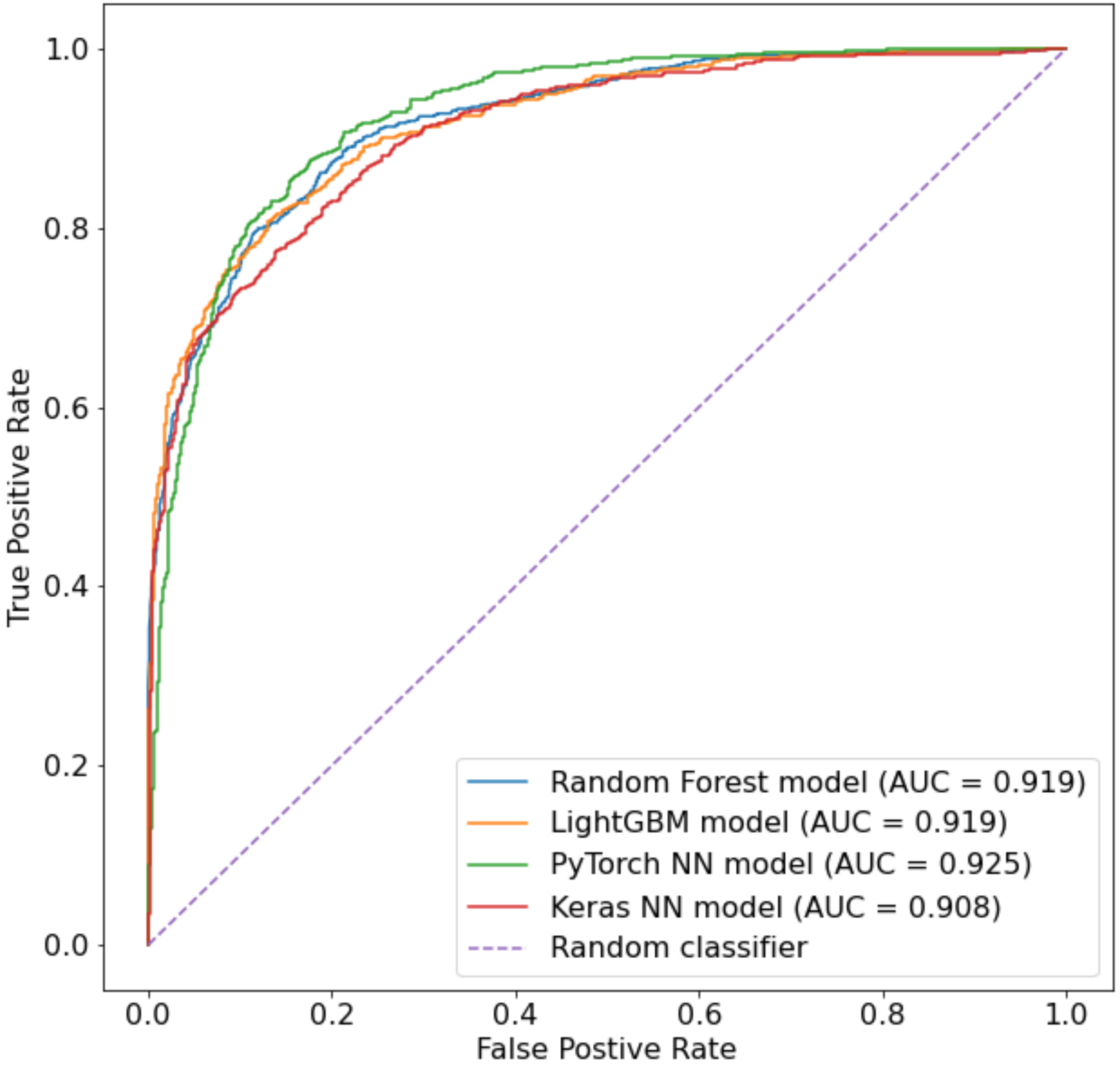
Accuracy	0.83
LogLoss	0.36
Fraction of wrong predictions	0.17
Area Under the ROC Curve	0.919

PyTorch Neural Network



Accuracy	0.85
LogLoss	0.35
Fraction of wrong predictions	0.15
Area Under the ROC Curve	0.925

Classification – finding false positives



Overall satisfying results from classification algorithms:

0.83	<	Accuracy	<	**0.85**
0.35	<	LogLoss	<	0.37
0.15	<	Wrong predictions	<	0.17
0.908	<	AUC	<	**0.925**

The Random Forest, LightGBM and Keras Neural Network present similar results.

PyTorch Neural network model gives the best results.

Regression – predicting planetary radii

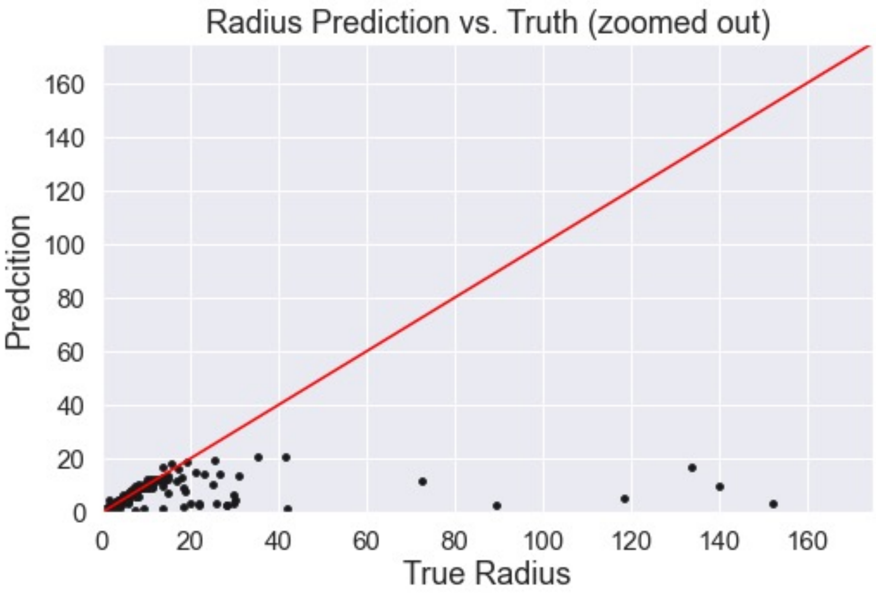
LightGBM model

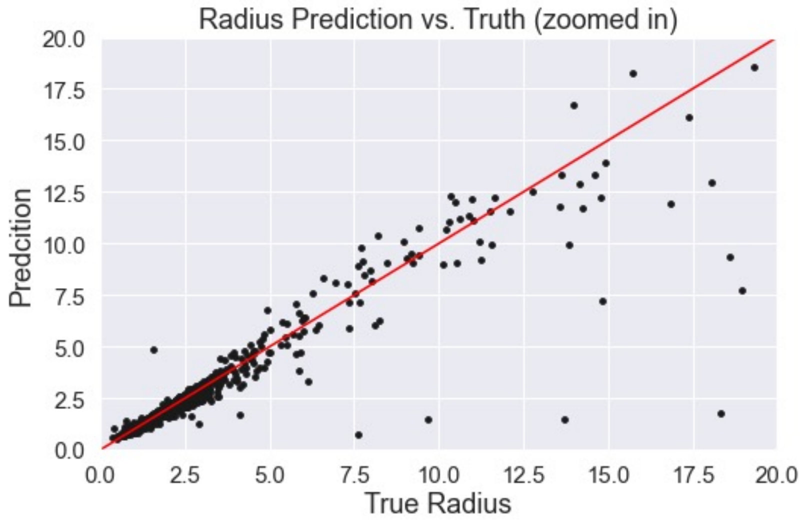
- performs well for small planets

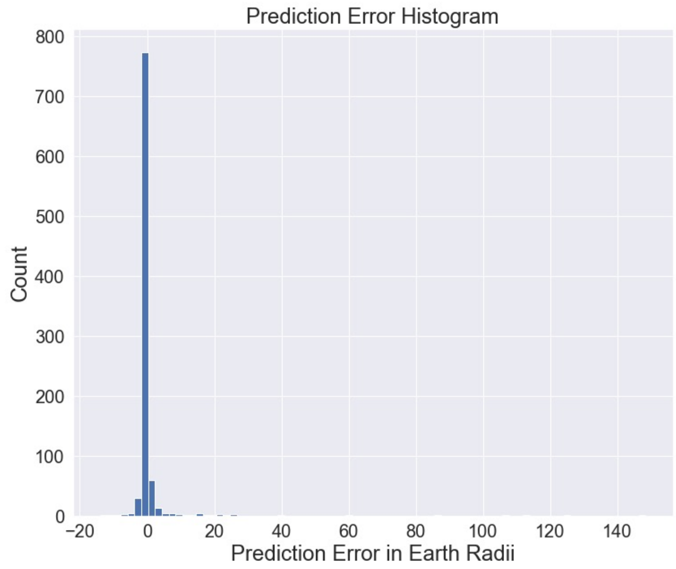
- Median absolute error: 0.007

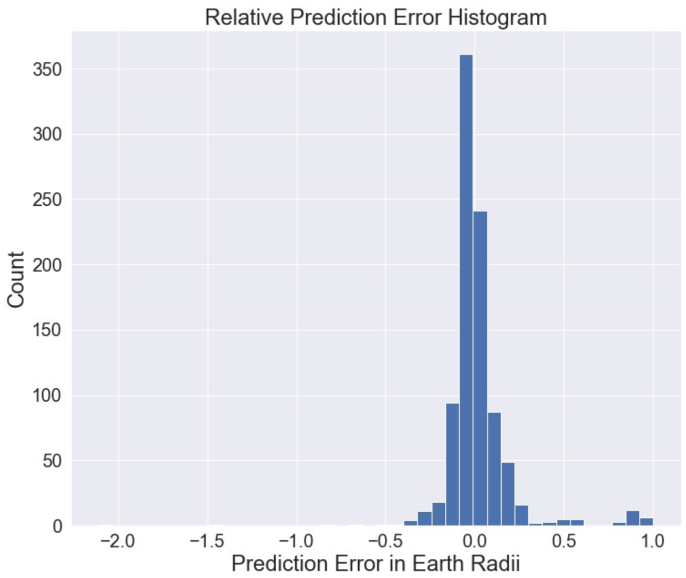
- few very large outliers

- one HUGE outlier (>100,000 Earth Radii) was ignored









Mean-Squared-Error ~ 92. 1
Mean-Absolute-Error ~ 1.4

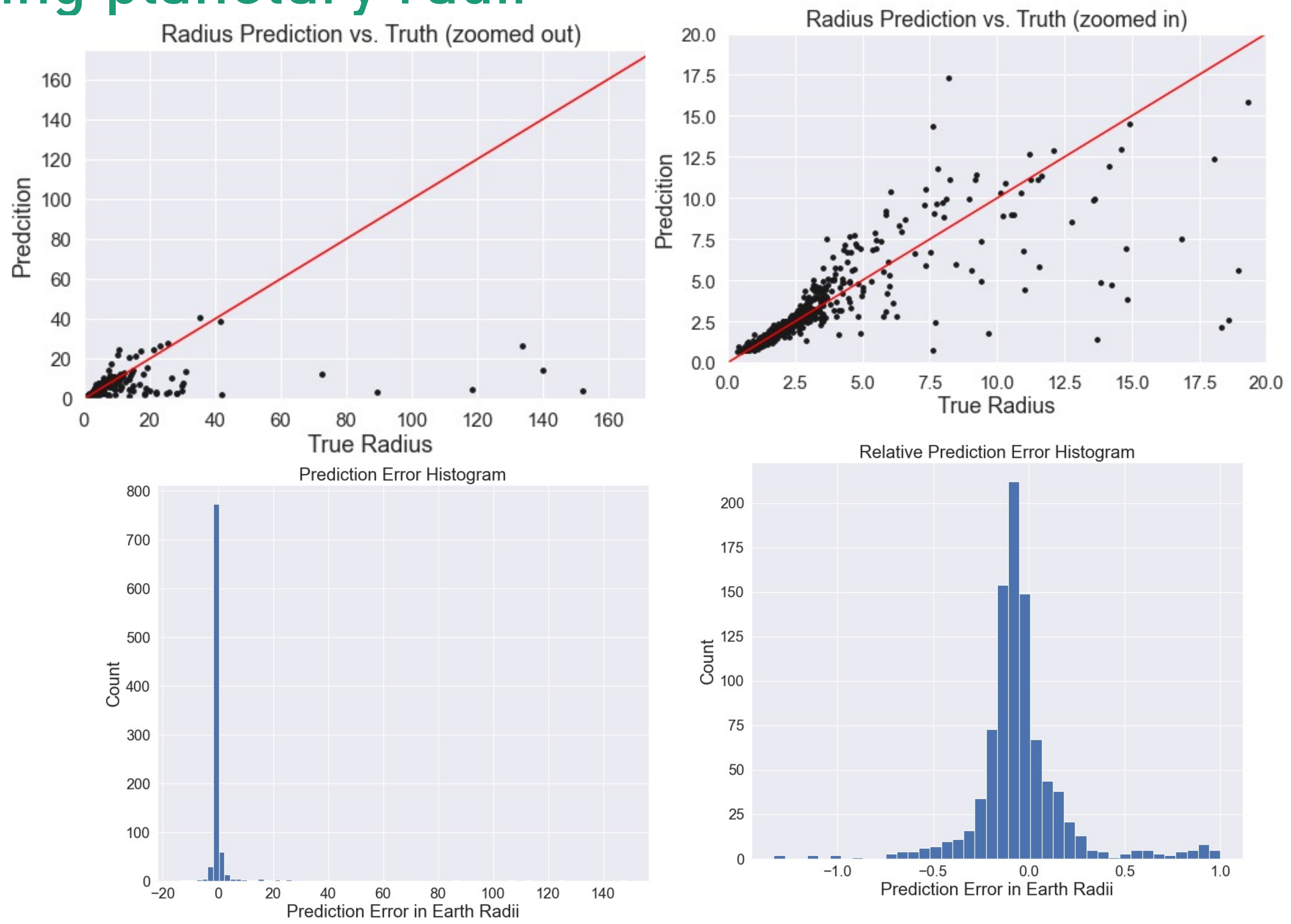
Regression – predicting planetary radii

Keras Neural Network model

- performs slightly worse than LightGBM model
- Median absolute Error: 0.14

Mean-Squared-Error ~ 94.0

Mean-Absolute-Error ~ 1.6

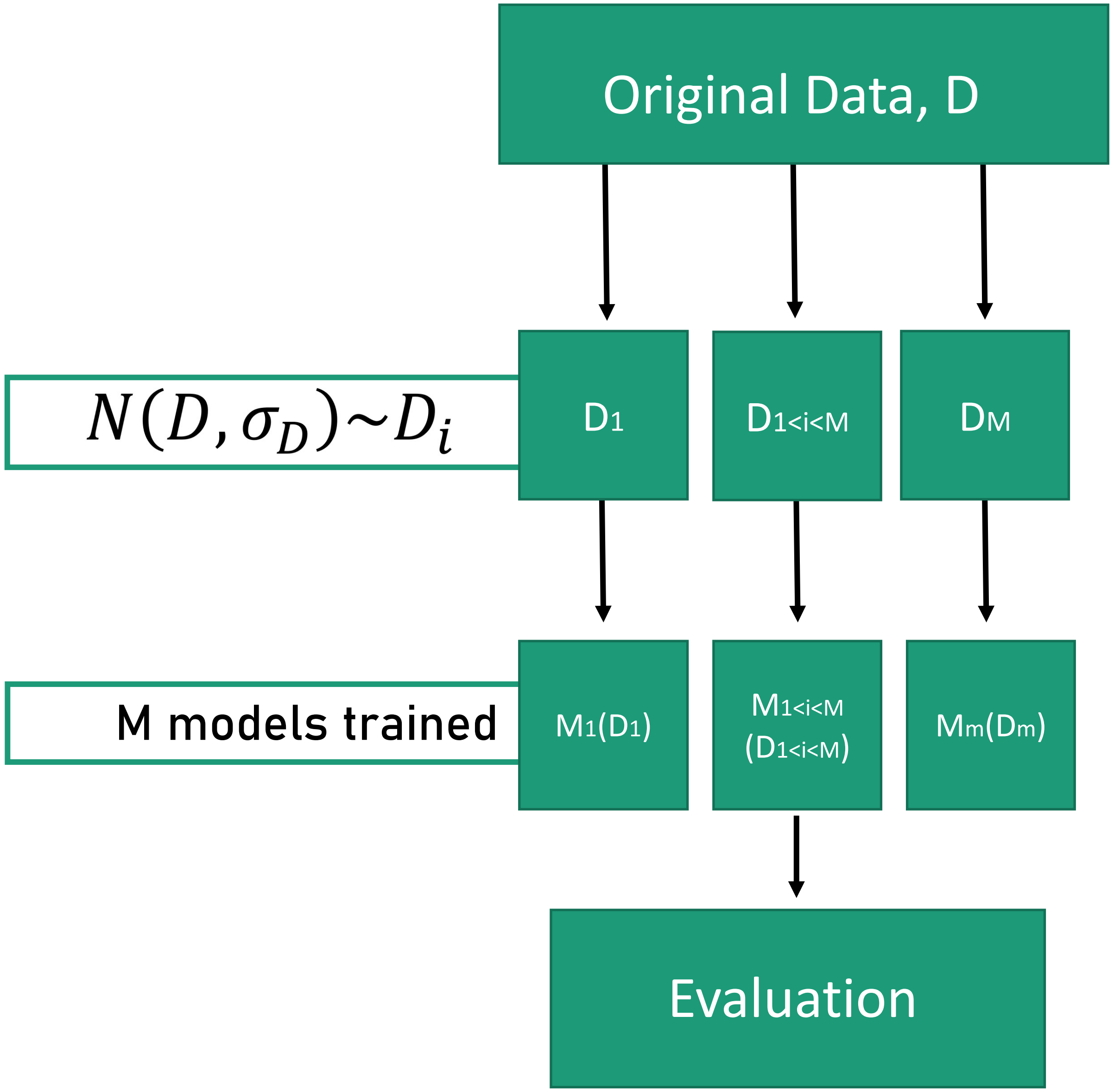


Data resampling

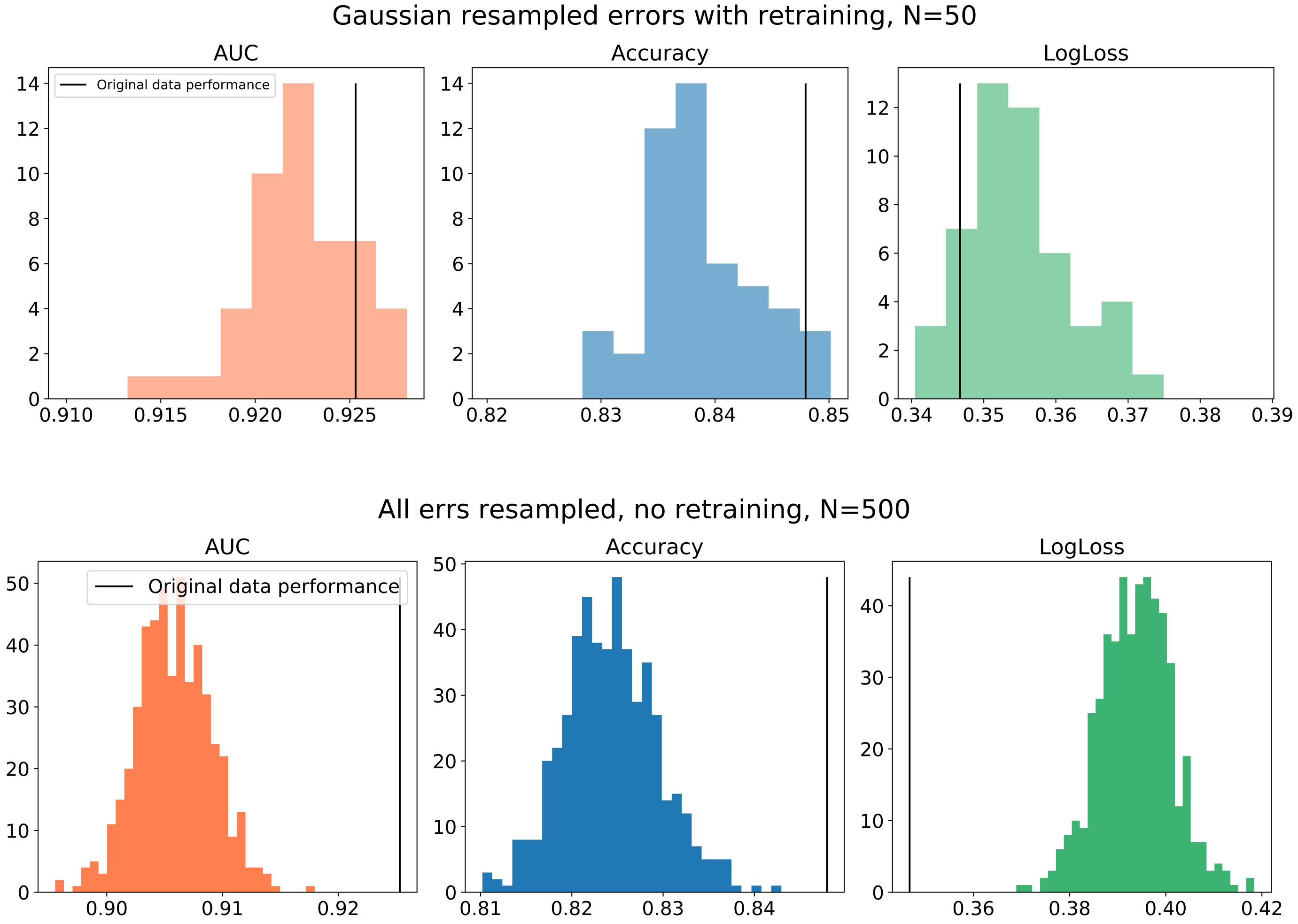
Our models should account for measurement errors.

Monte Carlo sampling is the answer.

Example for PyTorch Neural Network classification.



Data resampling



PyTorch NN	Original	No Retraining (N=500)	Retraining (N=50)
AUC	0.9253	0.9058±0.00014	0.9226±0.00041
Accuracy	0.8479	0.824±0.0002	0.8392±0.00075
LogLoss	0.3467	0.394±0.0003	0.354±0.001

Stacking Ensemble

Can a better model be constructed from en ensemble of different heterogenious models?

Best classification models give very similar predictions:

- KL divergence: 0.085
- Fischer's Correlation Coefficient: 0.93

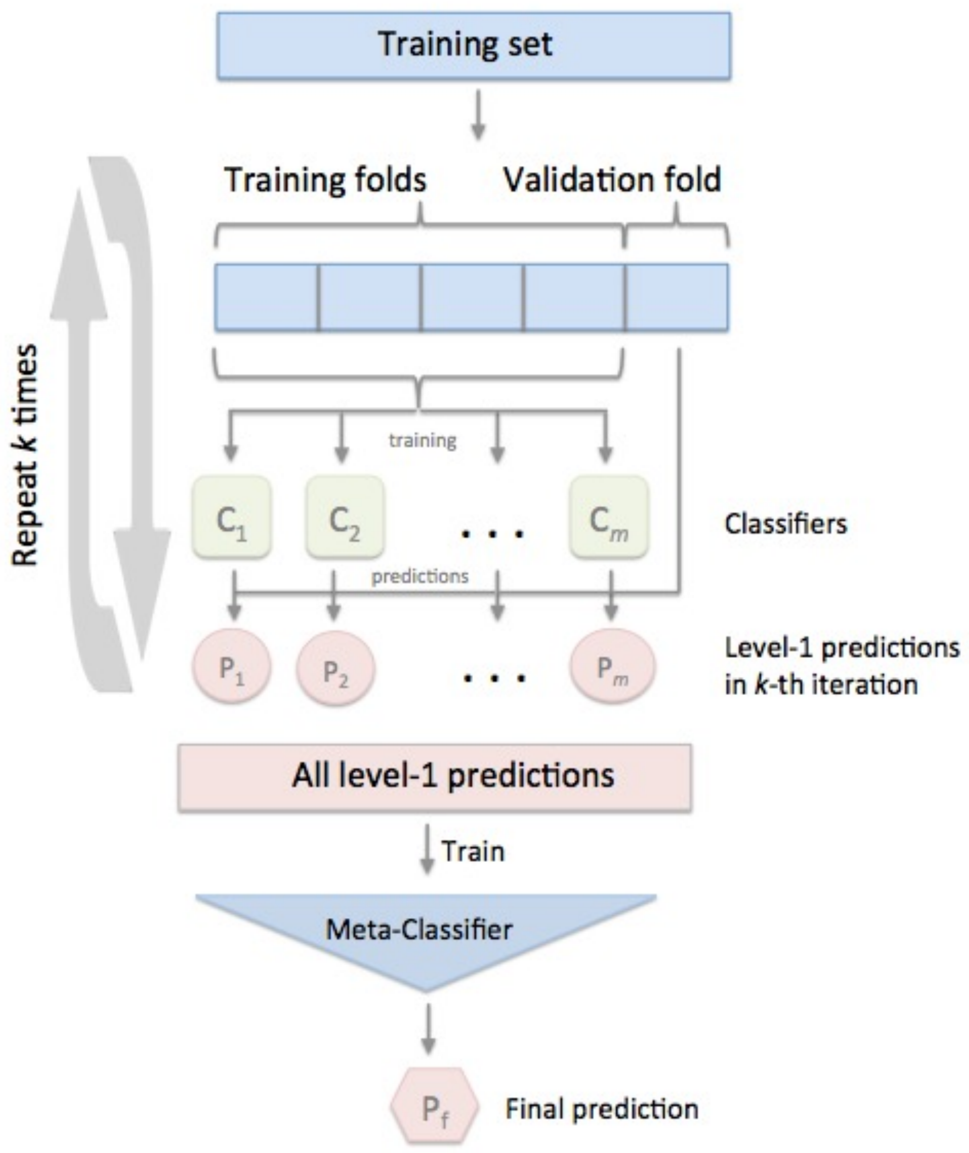


Image source: Tang, J., S. Alelyani, and H. Liu. "*Data Classification: Algorithms and Applications*." 2015

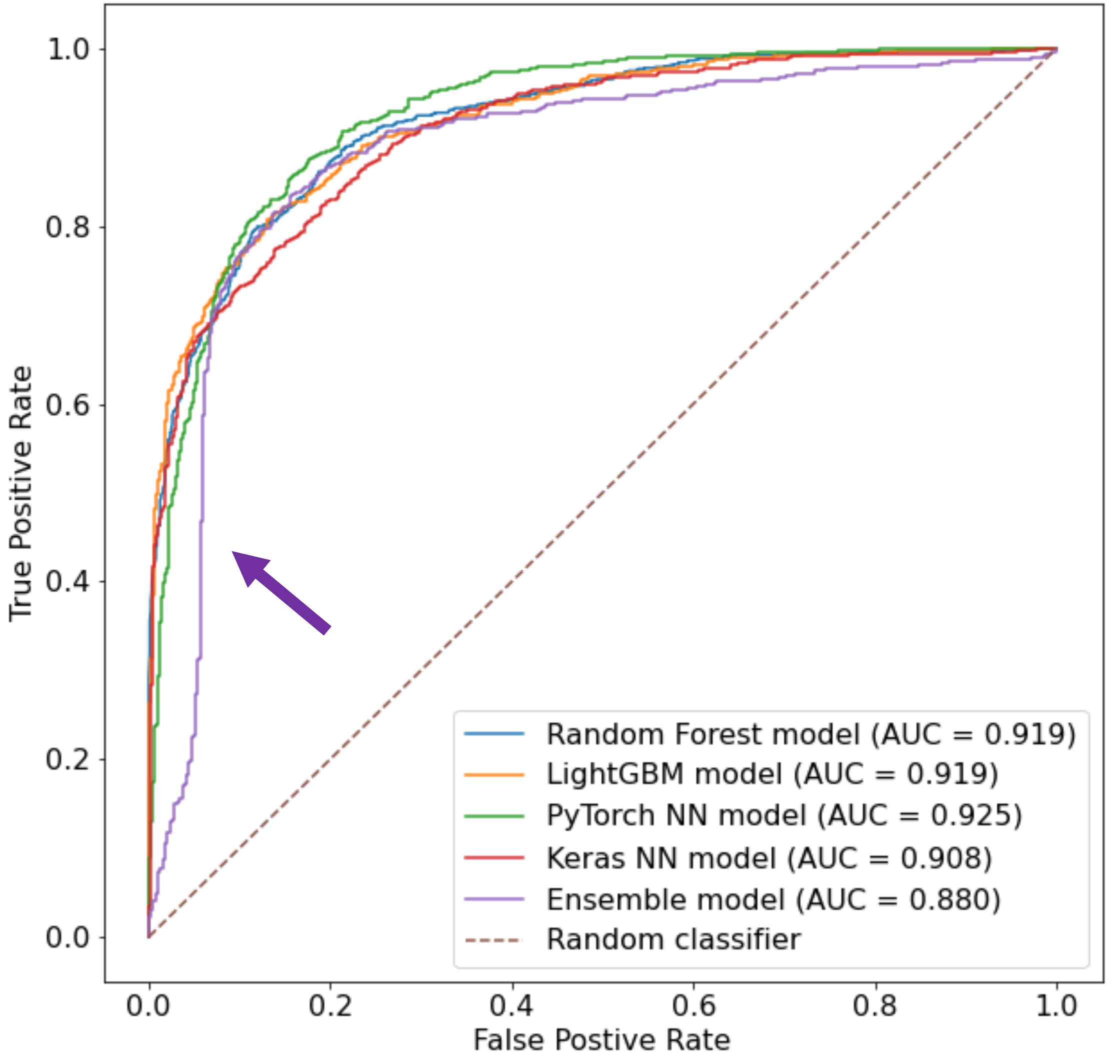
Stacking Ensemble

- Ensemble model types: Extreme Gradient Boost, Decision Tree, and kNN. Support Vector Machine as the meta-classifier.
- Ensemble requires more hyperparameter optimization, but theoretically the best way to combine our models.
- Ensemble performance:

    AUC = 0.880

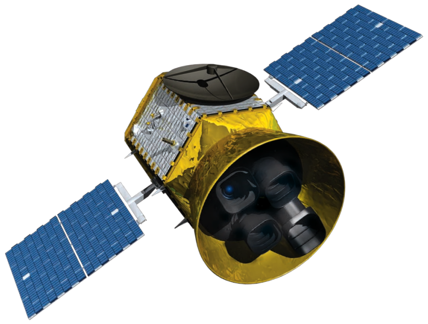
    Accuracy = 0.83

    LogLoss = 0.418

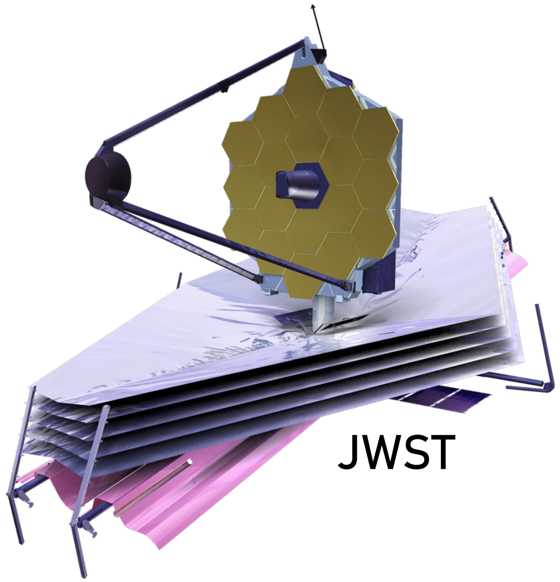


Summary and future work

- Models show fair performance, both for classification and regression.
- Results robust to adjusting for statistical errors.
- Ensemble methods ineffective when model predictions are similar.



TESS



JWST

Identifying Exoplanets with Deep Learning: A Five-planet Resonant Chain around Kepler-80 and an Eighth Planet around Kepler-90

Christopher J. Shallue[1] and Andrew Vanderburg[2,3,4]
[1] Google Brain, 1600 Amphitheatre Parkway, Mountain View, CA 94043, USA; shallue@google.com
[2] Department of Astronomy, The University of Texas at Austin, 2515 Speedway, Stop C1400, Austin, TX 78712, USA
[3] Harvard–Smithsonian Center for Astrophysics, 60 Garden Street, Cambridge, MA 02138, USA
Received 2017 September 19; revised 2017 November 13; accepted 2017 November 20; published 2018 January 30

Accurate Machine Learning Atmospheric Retrieval via a Neural Network Surrogate Model for Radiative Transfer

Michael D. Himes[1], Joseph Harrington[2], Adam D. Cobb[3], Atılım Güneş Baydin[3], Frank Soboczenski[4], Molly D. O'Beirne[5], Simone Zorzan[6], David C. Wright[1], Zacchaeus Scheffer[1], Shawn D. Domagal-Goldman[7] and Giada N. Arney[7]

[1] Planetary Sciences Group, Department of Physics, University of Central Florida
[2] Planetary Sciences Group, Department of Physics and Florida Space Institute, University of Central Florida
[3] Department of Engineering Science, University of Oxford
[4] SPHES, King's College London
[5] Department of Geology and Environmental Science, University of Pittsburgh
[6] ERIN Department, Luxembourg Institute of Science and Technology
[7] NASA Goddard Space Flight Center, Greenbelt, MD

Images source: *Wikipedia*

Appendix

Data overview/preprocessing

- The data was obtained from the NASA Exoplanet Archive. At this archive, one can find data from all NASA exoplanet hunting missions. The data is open source and very easy to download. Data here: https://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTbls&config=cumulative
- We choose to use the Kepler Objects of Interest (KOI) data. This includes all the data for Kepler's telescope first mission. KOI data includes false positives, confirmed as well as candidate planets that are yet to be confirmed to be planets.
- Some variables were excluded from the start – there were the variables we knew would have no influence on our models . These include variables like the planet IDs or the host star's positioning in the sky (from the observer's point of view).
- There weren't many missing variables, however we decided to exclude any entries which had missing parameters.
- In the final preprocessing we end up with 9200 data entries to build our models on.
- For all models we divide the train and test sets equally (20% test data) and with the same seed to ensure consistency when comparing the models.

The problems:

- Classification: The classification problem we try solve is to identify false positives in the KOI data.
- Regression: The regression problem we try to solve is to predict planetary radii in the KOI data.

Expectations:

- Classification: We expected the classification model to not perform as well as it did. This is because we were aware there are some outliers in the Kepler data which could make it hard to build solid models with an ok performance.
- Regression: We expected the regression to perform nicely because the detecting method is the best to measure planetary radii.

Classification:

Target variable: “Disposition Using Kepler Data” (as named on NASA exoplanet archive).

Features: Orbital period, transit duration, transit depth, planetary radius, insolation flux, planetary equilibrium temperature, stellar magnitude, stellar effective temperature, stellar surface gravity (log scale), stellar radius.

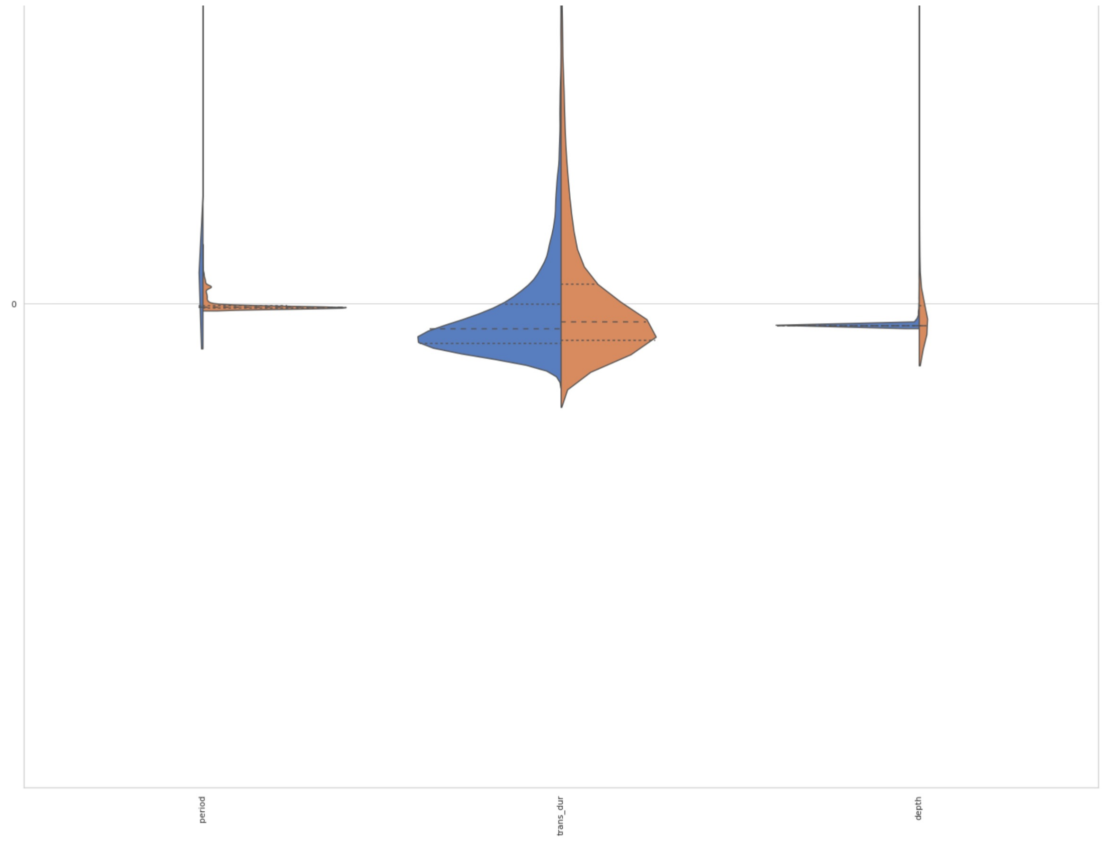
Regression:

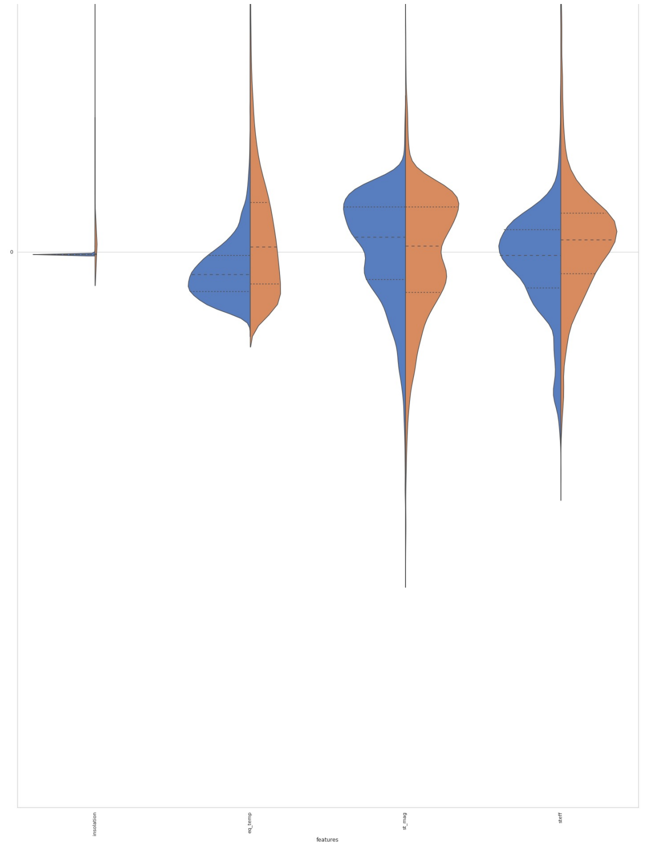
Target variable: Planetary Radius (in Earth radii).

Features: Orbital period, transit duration, transit depth, insolation flux, planetary equilibrium temperature, stellar magnitude, stellar effective temperature, stellar surface gravity (log scale), stellar radius.

Data analysis

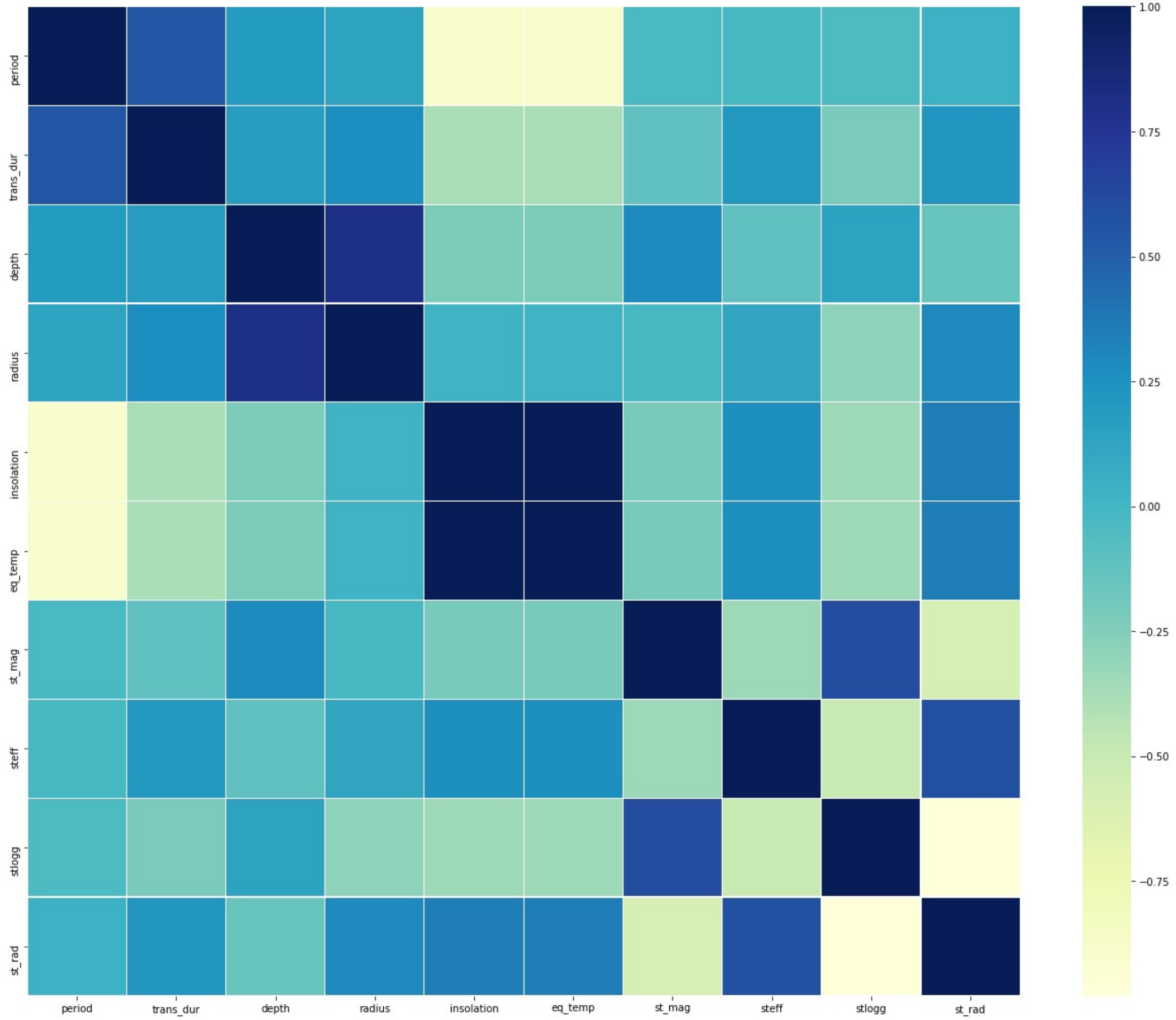
Data Analysis – Violin plots





Violin plots show the separation between candidate and false positive observations for the classification problem. For the majority of the variables, the mean is well separated which drives us to make an analysis using these variables as our features.

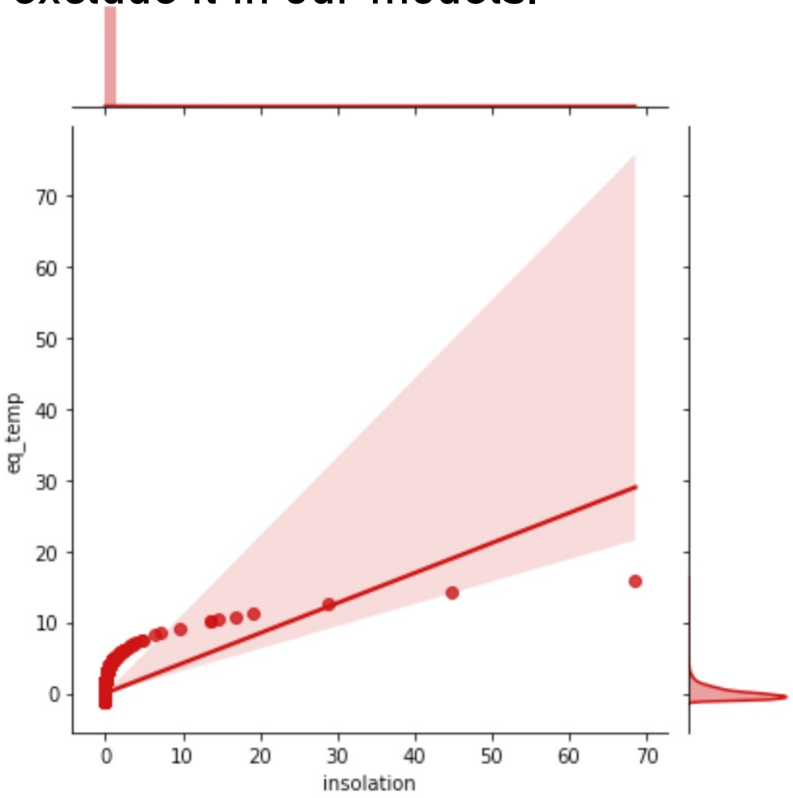
Data Analysis – correlation matrix

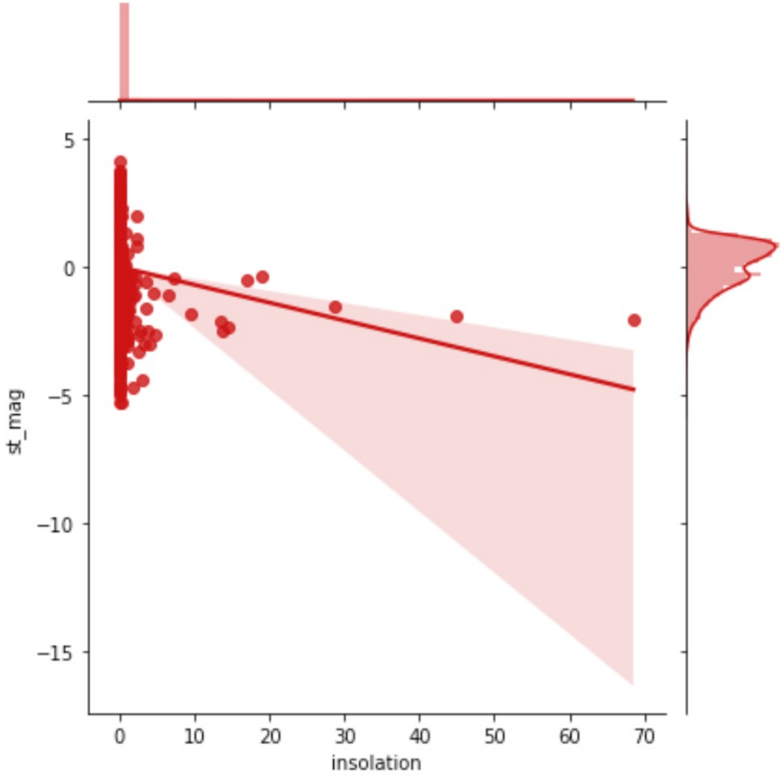


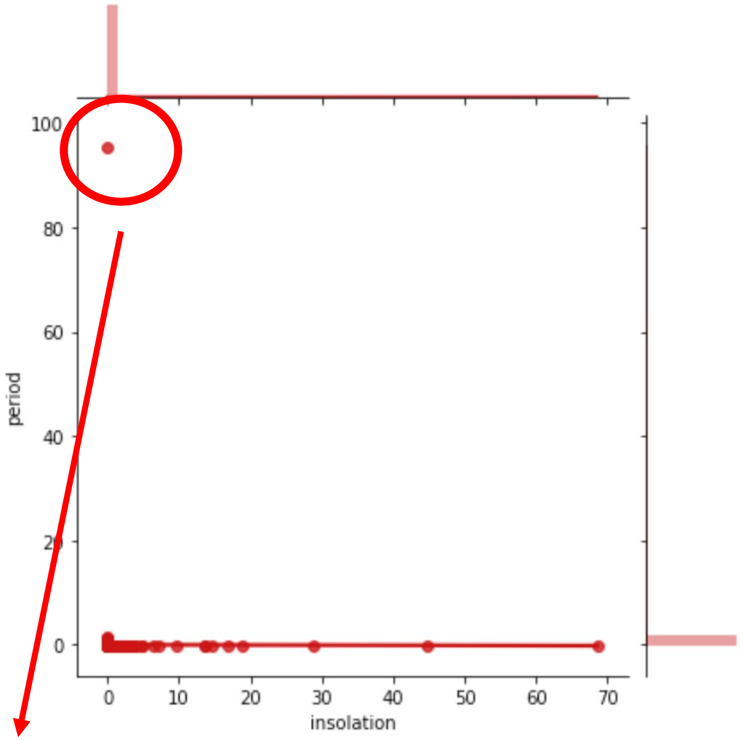
Using Spearman correlation we have identified which features have a high correlation and could be excluded in our models.
In particular the insolation flux and planetary equilibrium temperature show a very strong correlation (deep blue), which is physically expected.

Data analysis – 1vs1 feature correlation plots

For a deeper analysis of feature correlation, we used partial plots and compared each feature with the others. We have focused our effort on the insolation feature in order to understand if we could exclude it in our models.







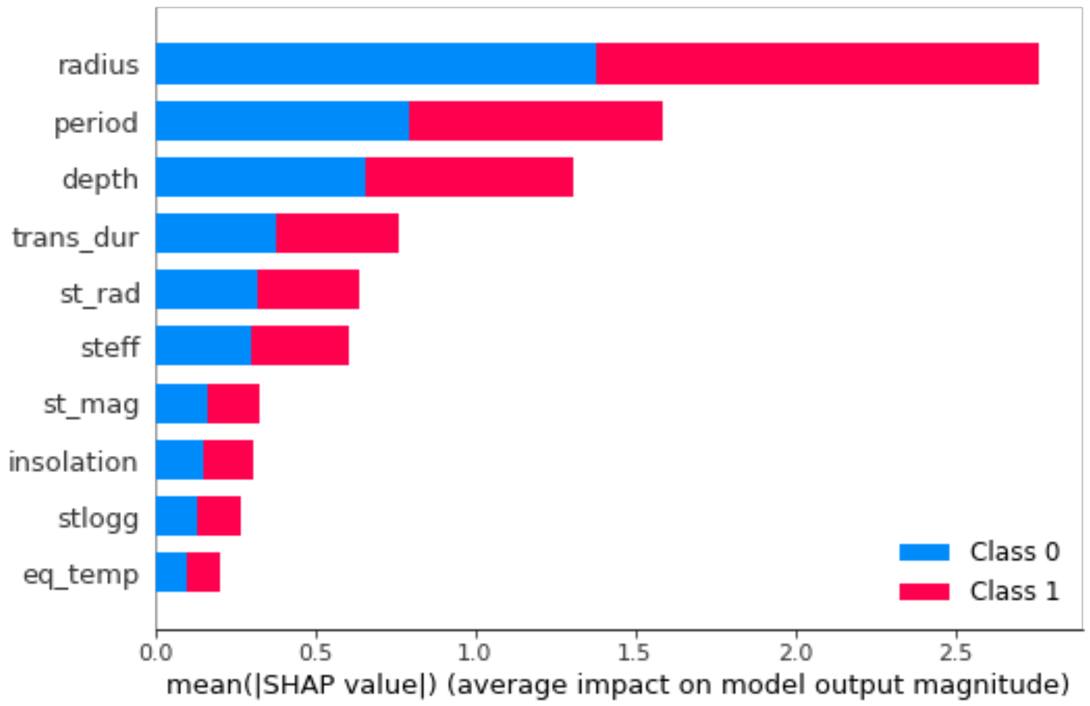
Note: outliers can have peculiar influence in this type of plot. The period is highly correlated to the insolation, however due to the outlier this does not seem to be the case just from the plot.

Classification models

LightGBM Classification

- Started by building a very simplified model with the parameters set to the defaults. The metric for the problem is the binary log loss. Used gbdt for the boosting type. With the simple model the log loss obtained was 0.372.
- Checked feature importance via shap values obtaining the results shown in the figure on the bottom right.
- The 8 most important features were chosen. This already excludes one of the two highly correlated features (the planet's equilibrium temperature is highly correlated to the insolation flux).
- Hyperparameterization was implemented with Optuna (includes Cross Validation). These are the hyperparameters:

```
params = {
    "objective": "binary",
    "metric": "binary_logloss",
    "verbosity": -1,
    "boosting_type": "gbdt",
    "feature_pre_filter": False,
    "lambda_l1": 3.1318293273194997,
    "lambda_l2": 1.388306806703893e-07,
    "num_leaves": 31,
    "feature_fraction": 0.9,
    "bagging_fraction": 0.99,
    "bagging_freq": 1,
    "min_child_samples": 20
    }
```

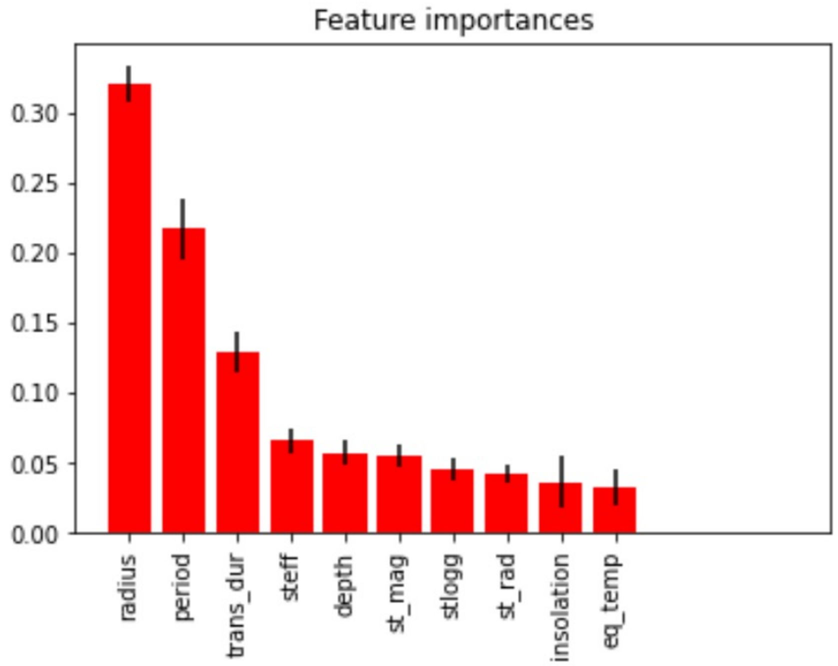


- Good performance overall however the neural networks were better
- LogLoss = 0.370; AUC = 0.919;
- The plotted ROC curve is in the presentation slides.

Random Forest classifier

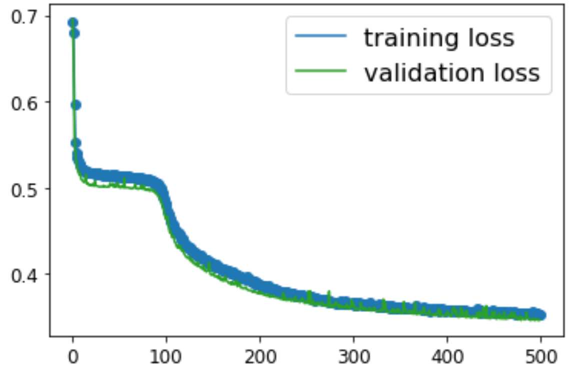
- Random Forest has been implemented. Features selected based on data analysis and Spearman correlation.
- Hyperparameters tuning performed with RandomSearch first, narrowing its results with a GridSearch as a final step.
- Feature importance has been calculated with importance based on mean decrease in impurity.
- Performance metrics to evaluate the model: LogLoss = 0.3594, accuracy = 0.829 and AUC= 0.919.

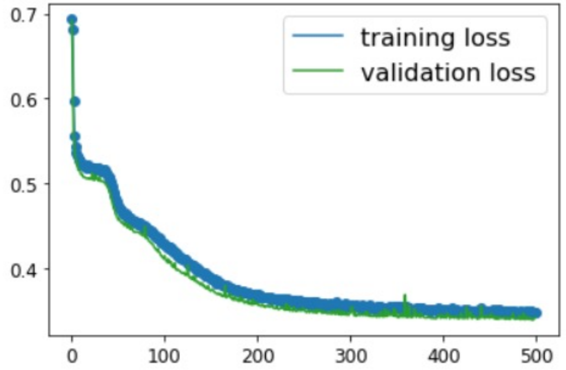
```
#random forest classifier
clf_rf = RandomForestClassifier(bootstrap=True, max_depth=20,
                                max_features=10,
                                n_estimators=300, n_jobs=None,
                                oob_score=False,
                                random_state=None, verbose=0,
                                warm_start=False)
clr_rf = clf_rf.fit(x_train,y_train)
```

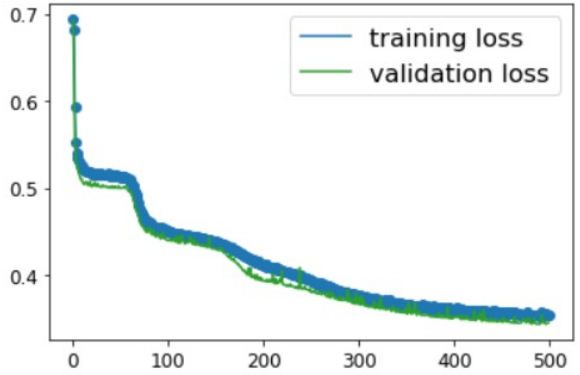


PyTorch feed forward NN for classification

- Sigmoid activation function
- Plateau in training curve is possibly local minimum/saddlepoint, (small gradient)
- Similar behaviour at different initial conditions, with various time spent at plateau.
- Training time could be reduced with eg. changing Lr
- All features used
- Preprocessed with Quantile scaler
- Binary Cross Entropy
- Training stopped when before validation loss increased.
- Grid search for different combinations of lr and epochs.
- Structure of the neural network:
  - 1 input layer
  - 2 hidden layers, 8 nodes each
  - 1 output layer
- Main hyperparameters:
  - learning_rate = 2e-3, batch_size = 25, n_epochs = 500







Keras NN Classification

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_layer (Dense)          (None, 13)                143
_________________________________________________________________
hidden_layer1 (Dense)        (None, 96)                1344
_________________________________________________________________
hidden_layer2 (Dense)        (None, 25)                2425
_________________________________________________________________
hidden_layer3 (Dense)        (None, 19)                494
_________________________________________________________________
hidden_layer4 (Dense)        (None, 25)                500
_________________________________________________________________
hidden_layer5 (Dense)        (None, 5)                 130
_________________________________________________________________
output (Dense)               (None, 1)                 6
=================================================================
Total params: 5,042
Trainable params: 5,042
Non-trainable params: 0
_________________________________________________________________
```

- Loss function: binary cross entropy
- Learning rate: 0.00095
- Density of the layers and lr optimised with kerastuner.

- Log loss: 0.366
- AUC: 0.908

Regression models

LightGBM Regressor

- Started by building a very simplified model with the following parameters:

```
params = {
    'boosting_type': 'gbdt',
    'objective': 'regression_l1',
    'metric': 'l1',
    'num_leaves': 30,
    'max_depth' : 10,
    'learning_rate': 0.01,
    'feature_fraction': 1.0,
    'bagging_fraction': 1.0,
    'bagging_freq': 1,
    'verbose': 0,
    'force_col_wise': True
}
```

- In this case, feature importance was not calculated and all variables available were used on the regressor model.
- Bayesian Optimization was used to optimize the max_depth, num_leaves and learning rate of the model. The hyperparameters after were:

```
params = {
    'boosting_type': 'gbdt',
    'objective': 'regression_l1',
    'metric': 'l1',
    'num_leaves': 601,
    'max_depth' : 866,
    'learning_rate': 0.006,
    'feature_fraction': 1.0,
    'bagging_fraction': 1.0,
    'bagging_freq': 1,
    'verbose': 0,
    'force_col_wise': True
}
```

- The model had a good performance overall; The results are comparable to the Keras Neural Network results.
- MAE = 119.53 – this is before we remove a very influencing outlier from the data set with an enormous radius.
- The plotted results are in the presentation slides.

Keras NN Regression

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_layer (Dense)          (None, 13)                130
_________________________________________________________________
hidden_layer1 (Dense)        (None, 96)                1344
_________________________________________________________________
hidden_layer2 (Dense)        (None, 25)                2425
_________________________________________________________________
hidden_layer3 (Dense)        (None, 19)                494
_________________________________________________________________
hidden_layer4 (Dense)        (None, 25)                500
_________________________________________________________________
hidden_layer5 (Dense)        (None, 5)                 130
_________________________________________________________________
output (Dense)               (None, 1)                 6
=================================================================
Total params: 5,029
Trainable params: 5,029
Non-trainable params: 0
_________________________________________________________________
```

- Loss function: MSE
- Learning rate: 0.00095
- Density of the layers and lr optimised with kerastuner.

- MSE without outlier: 94

Further work

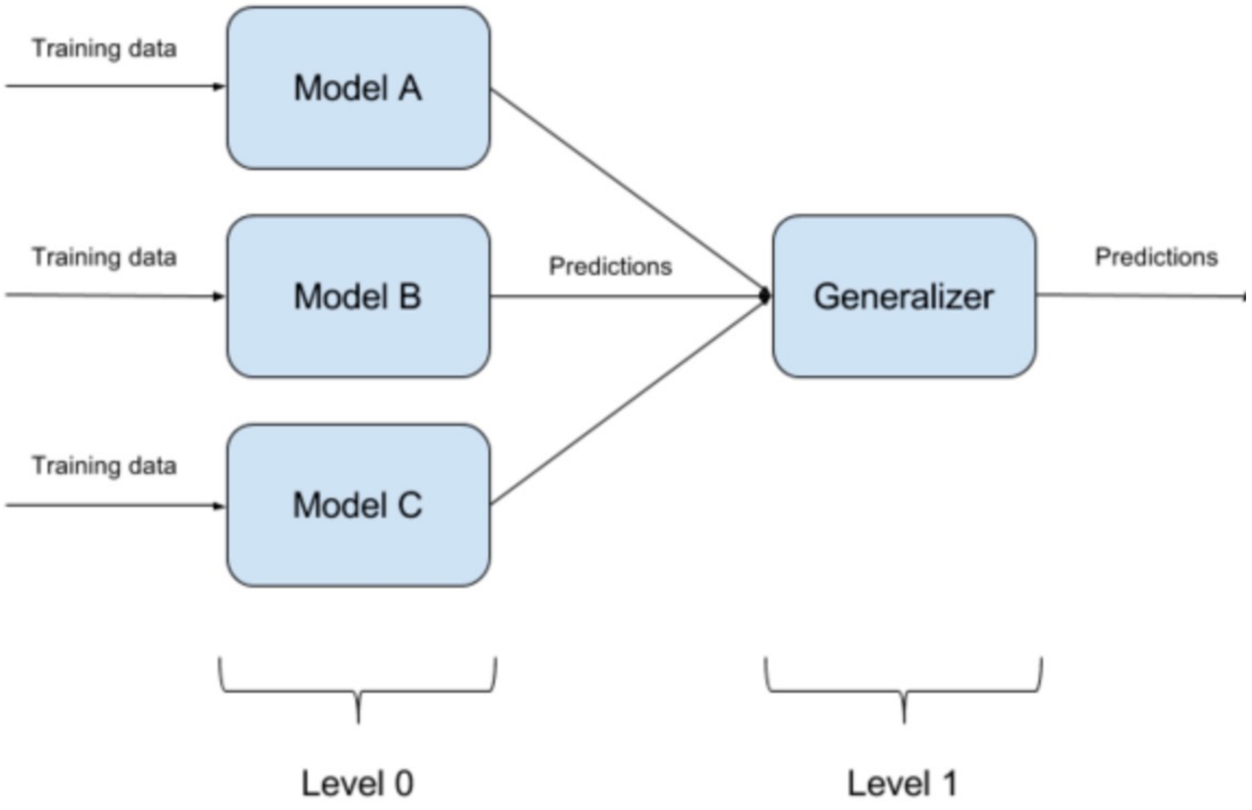
MC resampling

- Gaussian errors resampled within their standard deviances.
- This is strictly only a lower bound on the error, but we can't know the fit maximum likelihood landscapes.
- N=50 and N=500 was picked for computation time .
- Resampling was done with corr = 0, to mimic the fact that while features are correlated, measurement error is independent and a function of the measured feature.

Ensemble model – Level 0 choice

- An Ensemble model has been created with stacking architecture

- First step has been the development of several models to find best 3 classifiers to include in layer 0 based on accuracy:
- Random Forest = 83.7
- Decision Tree = 81.08
- Extreme Gradient Boost = 83.1
- K-nearest Neighbors = 75.1
- Support Vector Machine = 71.6



Ensemble model – Level 0 choice

- Hyperparameters tuning has
  been applied for the best 3 of the set.

- Xgboost: learning_rate=0.01, n_estimators
  =550, max_depth=20,gamma=0.6, subsam
  ple=0.52,colsample_bytree=0.6,seed=27, r
  eg_lambda=2,
  booster='dart', colsample_bylevel=0.6, col
  sample_bynode=0.5
- Sklearn DT: max_depth = 7

Ensemble model – Level 1 choice

- In order to choose level 1 model, we've compared the performances testing each model as level 1 choise
- Support Vector Machine has been the generalizer that gave better performances

Ensemble Model

- Stacking and metaclassifier optimized with mlxtend StackingCVClassifier
- Metaclassifier: Support Vector machine
- Models in ensemble: kNN, XGboost, sklearn.tree.
- Models hyperparameters relatively unoptimized

- Xgboost: learning_rate=0.01, n_estimators=550, max_depth=20,gamma=0.6, subsample=0.52,colsample_bytree=0.6,seed=27, reg_lambda=2, booster='dart', colsample_bylevel=0.6, colsample_bynode=0.5
- Sklearn DT: max_depth = 7
- Knn: N_neighbors = 300
- SVM: C=20

Other telescope data tests

- We tried to implement our models for the TESS Objects of Interest (TOI) data.
- TESS is a telescope that is currently hunting for exoplanets with the transit method. However, it is surveying brighter stars than Kepler did.
- The classification did not work. We believe this has to do with the fact that the stars observed by TESS are brighter and therefore the stellar parameters have a different range/scale.
- Curiously, the regression seems to not work so bad. It could be that the radius prediction is less dependent on stellar parameters (see figures on the right).