Inter IIT Tech Meet 2018 – IIT Madras
Event: Exoplanet Detection

ARCHITECTURE

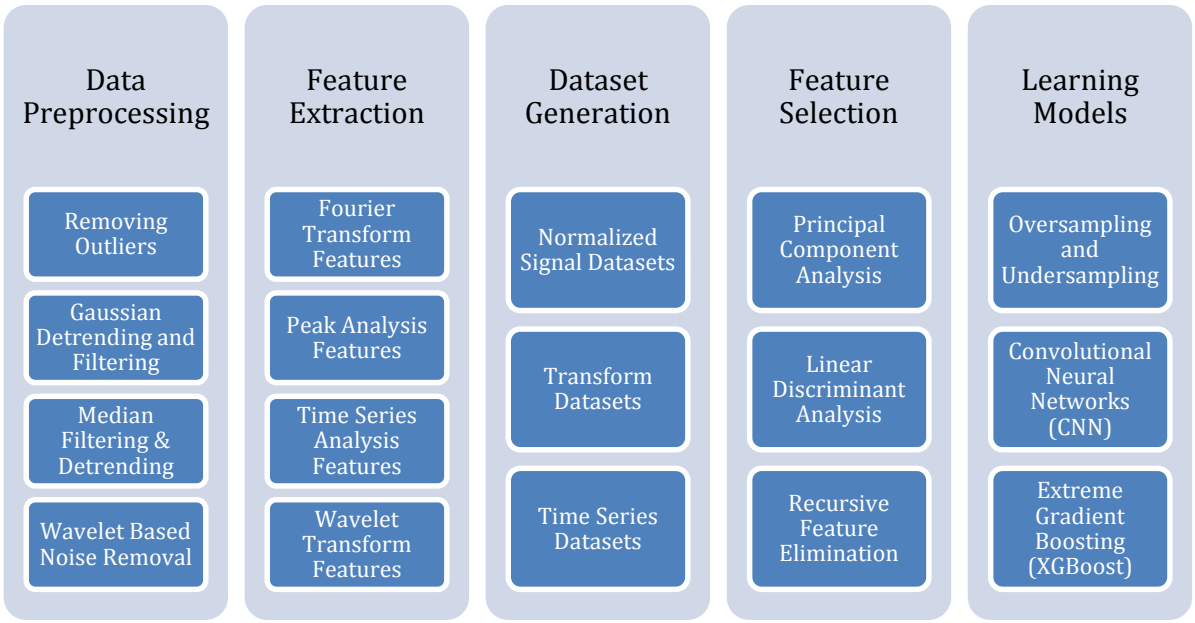

Figure 1: Architecture of Code

Github Repository: https://github.com/muditbac/ExoDet-Submission

METHODOLOGY

The dataset consists of a series of light intensities for 3960 stars, of which 33 have confirmed exoplanets around them. Visualizing the data indicated the presence of outliers, discontinuities[3], systematic trends, and excessive flux in the data which obscure the astral signals in the light curves. Some stars without exoplanets have dips at frequent intervals which could be due to phenomena like star spots, binary stars or stochastic noise. To deal with these, we process the raw data through our pipeline, which consists of the following steps.

Data Cleaning and Preprocessing

We started by removing outliers in the recorded signals. We removed only the upper outliers since lower outliers may indicate dips in flux intensities. We developed a hybrid algorithm for outlier detection, which identifies a data point as an outlier only if it is a local as well as a global anomaly (more in novelty section). After removing upper outliers, we used Gaussian filters and Uniform Median filters to eliminate systematic errors and low-frequency trends. We then removed high-frequency noise using Discrete Wavelet Transform.

Feature Extraction

After preprocessing, we extracted more than 7000 features from the data. We used time-series characteristics[2] such as autoregressive (AR) model parameters, autocorrelation, absolute energy, sample entropy, etc. as features. Using peak analysis[1], we obtained more features such as the number of downward peaks in the signal, mean and standard deviation of the peak heights and time differences between the peaks. We also extracted a time-series similarity feature using Dynamic Time Warping[2] (DTW), by calculating the similarity between a base signal and all other waveforms. Further, we used Fourier Transform and Wavelet Transform (more in novelty section) for alternate representations of the detrended signals. Finally, we compiled different datasets with different combinations of the above features along with normalized raw signals.

Machine Learning

Having formed datasets with a large number of features, we used dimensionality reduction techniques and feature selection techniques such as Recursive Feature Elimination (RFE), Principle Component Analysis (PCA) and Linear Discriminant Analysis (LDA) to reduce the overfitting and improve generalization of models. Since the dataset is highly imbalanced, we chose Area under Precision Recall Curve (AUPRC) over the AUC-ROC metric to evaluate our models. We also assessed the performance of models using F1-Score and True Skill Scores along with AUPRC. We calculated these metrics on 5-fold cross-validation predictions. While training our models, we made sure to avoid any data leakage in the system.

We initially experimented with different architectures of 1D Convolutional Neural Networks[4][5] (CNNs), on normalized signal datasets and Fourier transformed datasets. We found that the model with four convolutional layers outperformed the 25 layered Residual Network. Thus, the final architecture[4] contained four convolutional layers and three dense layers. We also trained Logistic Regression, K-Nearest Neighbors (KNN) and Support Vector Machines (SVM), but because of the high imbalance in the data, these models didn't perform well. We then applied Gradient Boosting classifiers on the dataset obtained from Wavelet transform and the time-series dataset with Peak and DTW features. To handle the high imbalance in the data, we used Undersampling and Oversampling techniques such as Edited Nearest Neighbors, All-KNN, and ADAYSN (Adaptive Synthetic Sampling). To combine the results from different models, we created an ensemble using a weighted average of the probabilities.

While tuning the models, we chose the model parameters that maximize the performance metrics on cross-validation predictions. To achieve this, we used the hyperopt package in Python, which picks values of the parameters based on Trees of Parzen Estimators. We have tuned all our models for around 800 hours of 16-core Intel Xeon CPU.

The table represents different models with their datasets along with their summaries.

Dataset	Model	F Score	Skill Score	AUPRC	AUC
Fourier Transform	PCA + XGB	0.5970	0.8212	0.4122	0.9610
Gaussian Detrended	PCA + XGB	0.6250	0.5722	0.4854	0.8228
Fourier Transform	UndSamp + PCA + XGB	0.6557	0.7698	0.5077	0.9497
Fourier Transform	CNN	0.6667	0.8513	0.5993	0.9628
Fourier Transform	PCA + XGB- Tune	0.6957	0.7568	0.4066	0.9387
Fourier Transform + AR Coeff	RFE + XGB	0.7273	0.8615	0.6257	0.9796
Gaussian Detrended	UndSamp + PCA + XGB	0.7273	0.7464	0.5795	0.9103
Fourier Transform	XGB	0.7761	0.9190	0.6299	0.9899
Normalized with Smoothed fit	CNN	0.7879	0.9287	0.7937	0.9962
Fourier Transform + AR Coeff	XGB	0.8000	0.8421	0.5845	0.9818
Normalized with Smoothed fit	CNN-Tune	0.8358	0.9353	0.8005	0.9873
Time Series Features	RFE + XGB	0.8525	0.8230	0.8230	0.9780
Normalized with Gaussian fit	CNN	0.8571	0.8174	0.8129	0.9816
Morlet Wavelet Transform	CNN	0.8615	0.9595	0.8328	0.9980
Time Series + Peak Features	RFE + XGB	0.8667	0.8424	0.8297	0.9859
Normalized with Gaussian fit	XGB	0.9206	0.8958	0.8563	0.9466
Discrete Wavelet Transform	XGB	0.9697	0.9694	0.9754	0.9991

Table 1: Cross-validation score of different models with different datasets

NOVELTY

Following are the novelties in our approach:

1) **Outlier Detection –** To identify local anomalies, our hybrid outlier detection algorithm fits a polynomial[6] to a sliding chunk of data, subtracts it to find the residuals, determines the standard deviations of the residuals, and identifies all the points with residuals further than a specified number of standard deviations from the fit. Similarly, to identify global anomalies, we used Gaussian filter and identified those points which are further than a specified number of standard deviations from the Gaussian-smoothed fit. After identifying the outliers, we replaced them with the median of their neighboring values.

2) **Wavelet Transform –** We removed high-frequency noise using the 2nd order Daubechies Wavelet Transform[7] by computing multilevel decomposition of the signal with an increased scale at every subsequent level. In this process, the downward dips are preserved. For feature extraction, we initially used Fast Fourier Transform (FFT) to decompose the detrended signals into frequencies. We found out that Fourier Transform does not represent abrupt changes efficiently because of the delocalization of sine waves in time. To extract an efficient representation of intensity dips, we used Wavelet Transform which is well localized in time and frequency resolutions. For this purpose, we applied Morlet Wavelet Transform on detrended signals and Daubechies Wavelet Transform on raw normalized signals.

3) **Training Convolutional Neural Networks –** Generally, CNNs are highly prone to overfitting. Thus, while training the CNNs we incorporated several data augmentation techniques such as random rolling, slicing, and flipping of the light curve signal. We found these data augmentation techniques to be highly effective in improving the performance of the CNNs as they helped the model to generalize to new examples. We also used Dropout (randomly omitting neurons) and Batch Normalization to enhance the training dynamics. We also used a smart batching technique while training, so as to ensure that there are equal number of positive and negative examples in each batch[4].

4) **Peak Analysis Features –** For peak analysis, we first identified peaks using sign changes in the first-order differences of the signal. Then we filtered out smaller peaks using thresholds for minimum peak gap and minimum peak height. Finally, we extracted the statistical features of peak gaps and heights, at different standard deviation levels below the mean.

REFERENCES

[1] Botros, Abraham. “Artificial Intelligence on the Final Frontier: Using Machine Learning to Find New Earths.” (2014).
[2] Ben D. Fulcher, Nick S. Jones: Highly comparative, feature-based time-series classification (2014)
[3] M Stumpe, J Smith, J Cleve, J Twicken, T Barclay, M Fanelli, F Girouard, J Jenkins, J Kolodziejczak ,S McCauliff and Robert L. Kepler Presearch Data Conditioning I—Architecture and Algorithms for Error Correction in Kepler Light Curves. Publications of the Astronomical Society of the Pacific, 2012
[4] Exoplanet Hunting in Deep Space, https://www.kaggle.com/keplersmachines/kepler-labelled-time-series-data
[5] Z Wang, W Yan and T Oates, Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline
[6] Python packages used: scikit-learn, pandas, numpy, hyperopt, xgboost, keras, mlextend, imblearn, PyWavelets, PyAstronomy
[7] KA Pearson, L Palafox, CA Griffith, Searching for Exoplanets using Artificial Intelligence