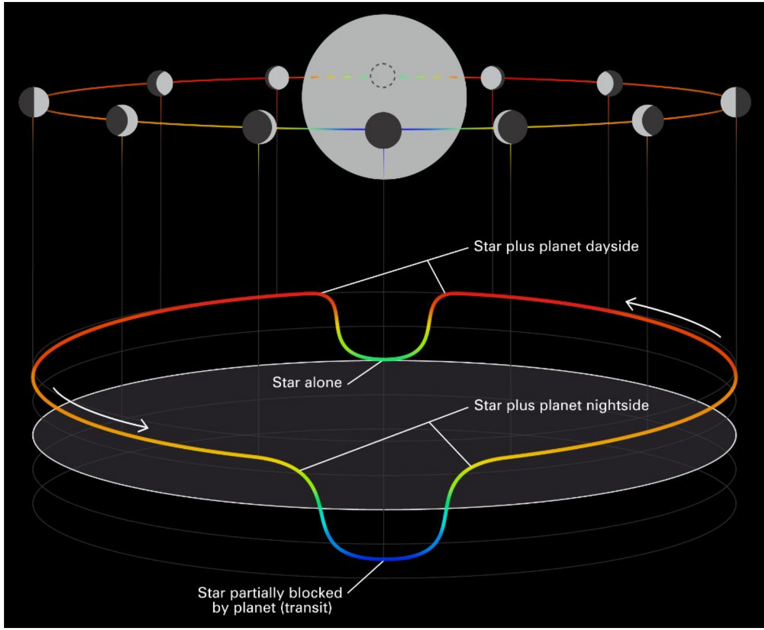
Q1 Machine Learning Project:

Prediction of Confirmed Status of Detected Exoplanet Candidates

Alan Zhu and Leonardo Valli

Introduction

- Exoplanets are planets orbiting stars other than our sun
- Currently ~5,800 confirmed exoplanets
- Confirming exoplanet candidates is time-consuming
- **Goal:** Predict confirmed exoplanet status using data about detected exoplanets



Dataset

- Transiting Exoplanet Survey Satellite (TESS) Dataset
- 61 attributes - identification, position, planet properties, stellar properties, dates.
- Class - TESS Follow-Up Working Group Disposition (TFOPWG): Confirmed or rejected
- https://exofop.ipac.caltech.edu/tess/

Attribute	**Meaning**
TOI	TESS Object of Interest number - unique identifier for the exoplanet candidate
Planet Orbital Period [days]	Time the planet takes to make a complete orbit around its host star.
Planet Transit Depth [ppm]	The relative flux (brightness) decrease caused by the orbiting body transiting in front of the star.
Planet Radius [R_Earth]	Radius of the planet.
TESS Magnitude	Brightness of the planet.
Stellar Radius [R_Sun]	Radius of the host star.

Preprocessing

Preprocessing

- Done in Python using Google Colab

>

Preprocessing - Instance Removal

- Remove instances of planets with no determined status
  - 7,000 → 2,000 instances
- Remove Instances with no class value

```
df.shape
```

```
(7217, 62)
```

```
df.dropna(subset=['TFOPWG Disposition'], inplace=True)#drop rows with missing class values
df.shape
```

```
(2164, 62)
```

```
df = df.drop(df[df['TFOPWG Disposition'] == 'PC'].index)
df = df.drop(df[df['TFOPWG Disposition'] == 'APC'].index)
df.shape
```

```
(2166, 62)
```

Preprocessing - Binary Encoding

- We made our class variable binary
  - Known Planet (KP) and Confirmed Planet (CP) → True
  - False Positive (FP) and False Alarm (FA) → False

```
[ ]  df['TFOPWG Disposition'].replace(['KP', 'FA'], ['CP', 'FP'], inplace=True)
     df['TFOPWG Disposition'].replace(['CP', 'FP'], ['True', 'False'], inplace=True)
```

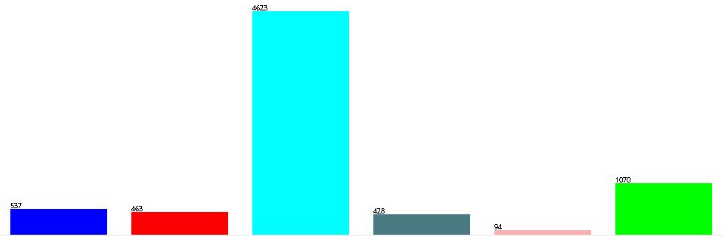
Selected attribute

Name: TFOPWG Disposition Type: Nominal

Missing: 2 (0%) Distinct: 6 Unique: 0 (0%)

No.	Label	Count	Weight
1	KP	537	537
2	CP	463	463
3	PC	4623	4623
4	APC	428	428
5	FA	94	94
6	FP	1070	1070

Class: TFOPWG Disposition (Nom) Visualize All



Preprocessing - Preliminary Attribute Removal

- Attribute removal: remove attributes we no intuitively have no predictive power
  - Identifiers
  - Positions
  - Dates
  - Error
  - 62 → 22

```
[ ]  df = df.drop(
         columns=[
             "TESS Mag err",
             "PM RA err (mas/yr)",
             "PM Dec err (mas/yr)",
             "Epoch (BJD) err",
             "Period (days) err",
             "Duration (hours) err",
             "Depth (mmag) err",
             "Depth (ppm) err",
             "Planet Radius (R_Earth) err",
             "Stellar Distance (pc) err",
             "Stellar Eff Temp (K) err",
             "Stellar log(g) (cm/s^2) err",
             "Stellar Radius (R_Sun) err",
             "Stellar Metallicity err",
             "Stellar Mass (M_Sun) err",
         ],
         axis=1,
     )  # columns that are just the error of other columns
```

```
df = df.drop(
    columns=[
        "TOI",
        "TIC ID",
        "Master",
        "SG1A",
        "SG1B",
        "SG2",
        "SG3",
        "SG4",
        "SG5",
        "Previous CTOI",
        "TESS Disposition",
        "Planet Name",
        "Pipeline Signal ID",
        "Source",
        "Detection",
        "RA",
        "Dec",
        "PM RA (mas/yr)",
        "PM Dec (mas/yr)",
        "Epoch (BJD)",
        "Sectors",
        "Date TOI Alerted (UTC)",
        "Date TOI Updated (UTC)",
        "Date Modified",
        "Comments",
    ],
    axis=1,
)  # columns that have no intuitive use for predictions
```

Preprocessing - Handling Missing Values

- Initially had 39072 missing values
- Remove columns with >75% missing
  - Stellar Metallicity
- Fill in with mean for columns with 0%<missing<75%

```
[ ] #deal with missing values

    for col in df.columns:
      num_missing = df[col].isnull().sum()

      percent_missing = (num_missing / len(df)) * 100

      print(f"Column '{col}': {num_missing} missing values, {percent_missing:.2f}% missing.")

      # If percentage of missing values is greater than 75%, drop the column
      if percent_missing > 75:
          df.drop(columns=[col], inplace=True)
          print(f"Column '{col}' dropped due to high percentage of missing values.")
      elif percent_missing > 0:
        mean_of_col = df[col].mean()
        df[col].fillna(mean_of_col, inplace=True)
        print(f"Missing values in column '{col}' filled with mean value {mean_of_col}")
```

Preprocessing - Normalization

- Z-score normalization
  - Presence of outliers

```
[ ] for col in df.drop(columns=['TFOPWG Disposition'], axis=1).columns:
      mean = df[col].mean()
      std = df[col].std()

      for row in df.index:
        df.loc[row, col] = (df.loc[row, col]-mean)/std

    df.head()
```

Preprocessing - Train test split

- Train-test-validation split: 70% / 15% / 15%
  - Stratified Random Sampling

```
from sklearn.model_selection import train_test_split

X = df.drop(columns = ['TFOPWG Disposition'], axis=1)
y = df['TFOPWG Disposition']

X_train, X_temp, y_train, y_temp = train_test_split(X, y, stratify=y, test_size=0.3)
X_test, X_val, y_test, y_val = train_test_split(X_temp, y_temp, stratify=y_temp, test_size=0.5)

X_train['TFOPWG Disposition'] = y_train
X_test['TFOPWG Disposition'] = y_test
X_val['TFOPWG Disposition'] = y_val
```

- Ended up using *k*-fold cross validation
  - Weka doesn’t support train-test-validation split

Us when Weka didn't take our validation set

- test.arff
- test-cfs_subset.arff
- test-correlation.arff
- test-gain_ratio.arff
- test-reliefF.arff
- test-self_selected.arff
- TOI.arff
- TOINORMALIZED.arff
- train.arff
- train-cfs_subset.arff
- train-correlation.arff
- train-gain_ratio.arff
- train-reliefF.arff
- train-self_selected.arff
- validation.arff
- validation-cfs_subset.arff
- validation-correlation.arff
- validation-gain_ratio.arff
- validation-reliefF.arff
- validation-self_selected.arff

Attribute Selection

Correlation

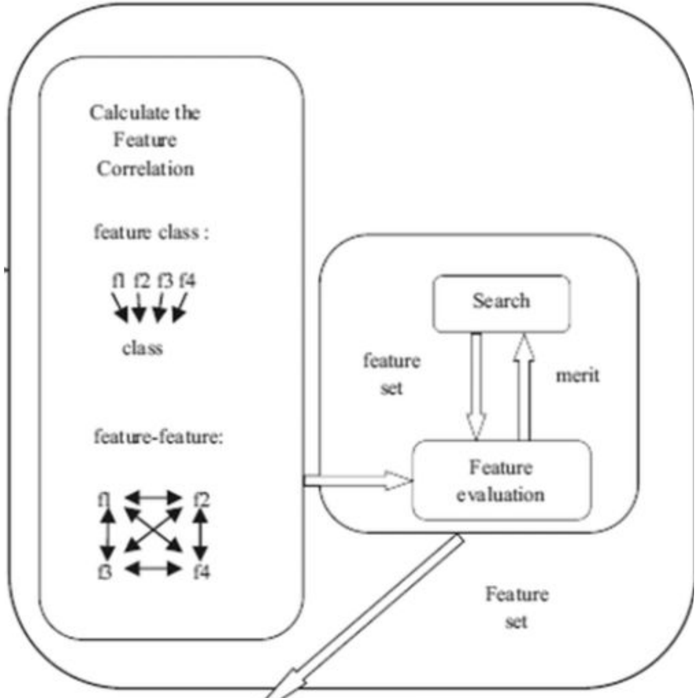
- Pearson Correlation Coefficient

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

```
Ranked attributes:
 0.3761   14 Planet Equil Temp (K)
 0.3109   20 Stellar Mass (M_Sun)
 0.2617   17 Stellar Eff Temp (K)
 0.2602   18 Stellar log(g) (cm/s^2)
 0.2548    6 Imaging Observations
 0.2363    4 Time Series Observations
 0.2334    5 Spectroscopy Observations
 0.1977   16 Stellar Distance (pc)
 0.1948   13 Planet Insolation (Earth Flux)
 0.1399   19 Stellar Radius (R_Sun)
 0.1342   15 Planet SNR
 0.0877   12 Planet Radius (R_Earth)
 0.0653    9 Duration (hours)
 0.0409   11 Depth (ppm)
 0.04      2 TSM
 0.0384    3 Predicted Mass (M_Earth)
 0.0296   10 Depth (mmag)
 0.0185    1 ESM
 0.0143    8 Period (days)
 0.012     7 TESS Mag
```

CFS Subset

- Correlation-based Feature Subset
- Evaluates intercorrelation
- 12 attributes



4,5,6,8,10,12,13,14,15,16,18,20 : 12
Time Series Observations
Spectroscopy Observations
Imaging Observations
Period (days)
Depth (mmag)
Planet Radius (R_Earth)
Planet Insolation (Earth Flux)
Planet Equil Temp (K)
Planet SNR
Stellar Distance (pc)
Stellar log(g) (cm/s^2)
Stellar Mass (M_Sun)

Gain Ratio

- Information gain (entropy)
- Expected value of information
- Probability distribution

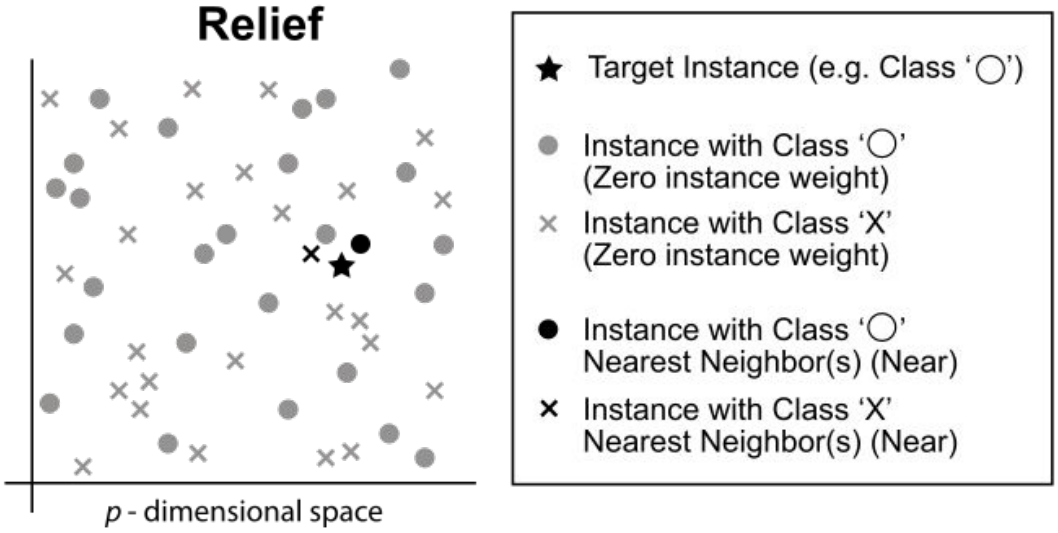
$$H(P) = -\sum_{x \in C} P(x) \log P(x)$$

(Shannon entropy)

0.1125	13	Planet Insolation (Earth Flux)
0.0846	14	Planet Equil Temp (K)
0.0823	4	Time Series Observations
0.0786	20	Stellar Mass (M_Sun)
0.0771	18	Stellar log(g) (cm/s^2)
0.0621	12	Planet Radius (R_Earth)
0.0609	19	Stellar Radius (R_Sun)
0.0592	17	Stellar Eff Temp (K)
0.0555	16	Stellar Distance (pc)
0.0519	10	Depth (mmag)
0.0519	11	Depth (ppm)
0.0505	8	Period (days)
0.0444	5	Spectroscopy Observations
0.0409	15	Planet SNR
0.0385	6	Imaging Observations
0.0327	1	ESM
0.0324	3	Predicted Mass (M_Earth)
0.0222	7	TESS Mag
0	9	Duration (hours)
0	2	TSM

Relief

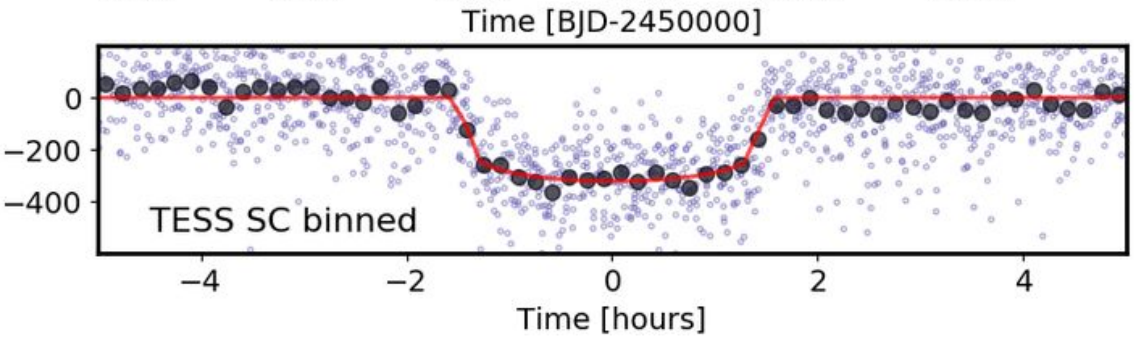
- Calculate feature score
- Closest same-class and different-class neighbors
- Association between attribute and class and inter-attribute relationships




```
Ranked attributes:
 0.027226   14 Planet Equil Temp (K)
 0.015745   20 Stellar Mass (M_Sun)
 0.011189    4 Time Series Observations
 0.010021    7 TESS Mag
 0.008191   18 Stellar log(g) (cm/s^2)
 0.007213    5 Spectroscopy Observations
 0.006352    9 Duration (hours)
 0.006173    6 Imaging Observations
 0.005643   13 Planet Insolation (Earth Flux)
 0.004976   17 Stellar Eff Temp (K)
 0.004535   16 Stellar Distance (pc)
 0.003795    3 Predicted Mass (M_Earth)
 0.002523   15 Planet SNR
 0.002195    1 ESM
 0.001401   11 Depth (ppm)
 0.00132    12 Planet Radius (R_Earth)
 0.00099    10 Depth (mmag)
 0.000871   19 Stellar Radius (R_Sun)
 0.000841    2 TSM
 0.000662    8 Period (days)
```

Self-Selected

- Planetary characteristics
- Stellar characteristics
- Significance (depth, SNR)
- Number of observations




```
Depth (ppm)
Planet Radius (R_Earth)
Planet Insolation
Planet Equil Temp (K)
Planet SNR
Stellar Eff Temp (K)
Stellar log(g) (cm/s^2)
Stellar Radius (R_Sun)
Stellar Mass (M_Sun)
Time Series Observations
Spectroscopy Observations
Imaging Observations
```

Classification

Classification - OneR

```
For every attribute in the training data:
     For every value for that attribute:
          For every class value:
               Calculate error for the rule value -> class value
          Remember the best performing rule for this value
     Remember the best performing rules for this attribute
Final rules = best performing rules from one attribute
```

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3

IF Outlook = Sunny THEN PlayGolf = Yes
IF Outlook = Overcast THEN PlayGolf = Yes
IF Outlook = Rainy THEN PlayGolf = No

Classification - NaiveBayes

```
For every class value:
     Find percentage of training data that is that class
     For every attribute:
          For every value of that attribute:
               Calculate likelihood of that class having that value

Classification = max(percentage * likelihoods)
```

Likelihood of the Evidence given that the Hypothesis is True

Prior Probability of the Hypothesis

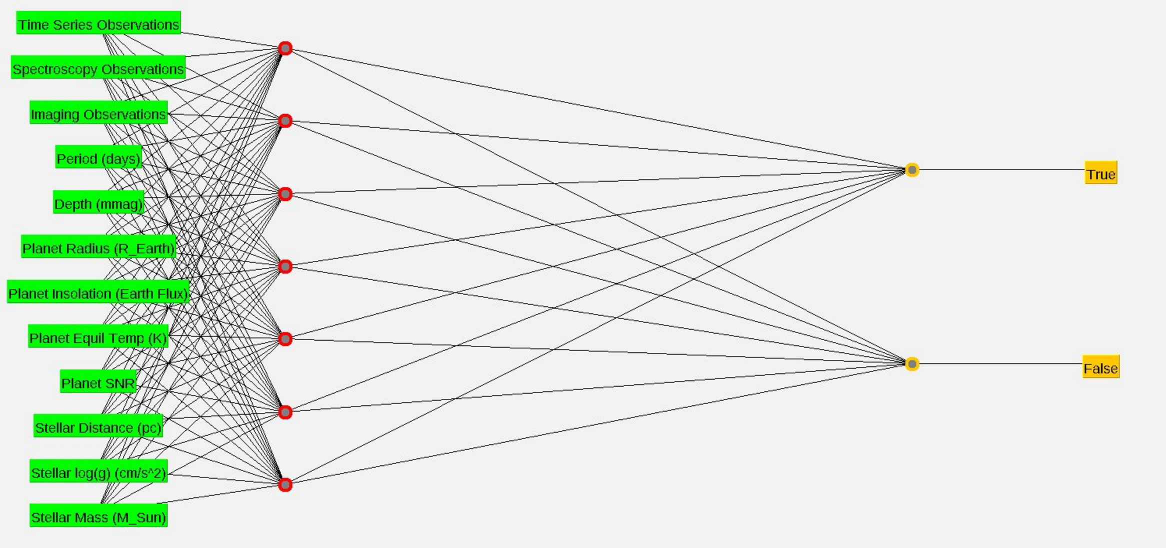
$$P(H|E) = \frac{P(E|H) * P(H)}{P(E)}$$

Posterior Probability of the Hypothesis given that the Evidence is True

Prior Probability that the evidence is True

Classification - MultilayerPerceptron

```
Initialize random base weights for network
For every instance in the training set:
    Make prediction using current weights
    Calculate loss (e.g., Mean Squared Error) by comparing prediction to actual class
    For every weight in the NeuralNet:
        Update via gradient descent
```

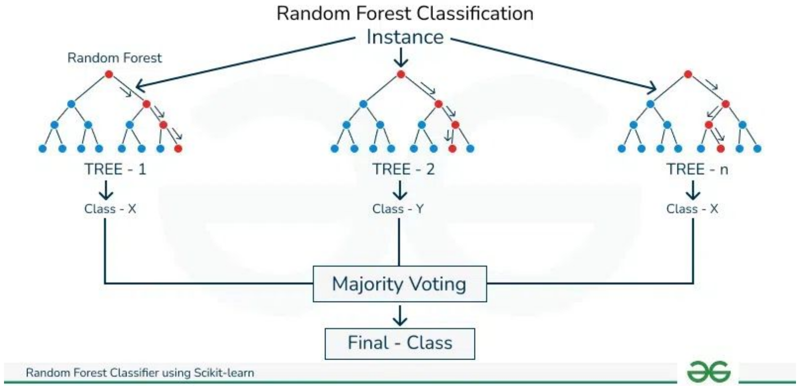


Classification - J48

```
Planet Equil Temp (K) <= -0.024019
|   Stellar Distance (pc) <= -0.543783
|   |   Depth (mmag) <= 0.005916
|   |   |   Planet SNR <= -0.37245: False (2.0)
|   |   |   Planet SNR > -0.37245: True (51.0/10.0)
|   |   Depth (mmag) > 0.005916: False (5.0/1.0)
|   Stellar Distance (pc) > -0.543783: False (300.0/90.0)
Planet Equil Temp (K) > -0.024019: False (416.0/26.0)
```

Classification - RandomForests

```
For i in range(n):
    Create decision tree
For i in range(n): #Classification
    Have the i-th decision tree make a prediction on the test instance
Classification = Class with the most votes from the decision trees
```



Results

Results - Accuracies

Accuracy

		Attribute Selection Method				
		Correlation Attribute Eval	Gain Ratio Attribute Eval	ReliefF Attribute Eval	Cfs Subset Eval	Our Choice of Attribute Subset
Classification Method	OneR	66.54%	66.54%	67.05%	66.54%	66.54%
	NaiveBayes	65.53%	63.22%	67.51%	62.80%	65.99%
	RandomForests	87.38%	86.65%	86.83%	$\boxed{87.89\%}$	86.46%
	MultilayerPerceptron	79.99%	79.02%	82.39%	80.73%	80.45%

Results - Area under ROC Curve

ROC Area

		Attribute Selection Method				
		Correlation Attribute Eval	Gain Ratio Attribute Eval	ReliefF Attribute Eval	Cfs Subset Eval	Our Choice of Attribute Subset
Classification Method	OneR	0.665	0.665	0.670	0.665	0.665
	NaiveBayes	0.815	0.803	0.801	0.803	0.820
	RandomForests	0.945	0.937	0.941	0.949	0.943
	MultilayerPerceptron	0.887	0.869	0.886	0.884	0.881

Results - Confusion Matrices

Correlation Attribute Eval

OneR

		Predicted	
		True	False
Actual	True	663	337
	False	387	777

NaiveBayes

		Predicted	
		True	False
Actual	True	912	88
	False	658	506

RandomForests

		Predicted	
		True	False
Actual	True	873	127
	False	146	1018

MultilayerPerceptron

		Predicted	
		True	False
Actual	True	794	206
	False	227	937

Gain Ratio Attribute Eval

OneR

		Predicted	
		True	False
Actual	True	663	337
	False	387	777

NaiveBayes

		Predicted	
		True	False
Actual	True	925	75
	False	721	443

RandomForests

		Predicted	
		True	False
Actual	True	871	129
	False	160	1004

MultilayerPerceptron

		Predicted	
		True	False
Actual	True	796	204
	False	250	914

ReliefF Attribute Eval

OneR

		Predicted	
		True	False
Actual	True	665	335
	False	378	786

NaiveBayes

		Predicted	
		True	False
Actual	True	863	137
	False	566	598

RandomForests

		Predicted	
		True	False
Actual	True	867	133
	False	152	1012

MultilayerPerceptron

		Predicted	
		True	False
Actual	True	854	146
	False	235	929

Cfs Subset Eval

OneR

		Predicted	
		True	False
Actual	True	663	337
	False	387	777

NaiveBayes

		Predicted	
		True	False
Actual	True	906	94
	False	711	453

RandomForests

		Predicted	
		True	False
Actual	True	873	127
	False	135	1029

MultilayerPerceptron

		Predicted	
		True	False
Actual	True	808	192
	False	225	939

Our Choice of Attribute Subset

OneR

		Predicted	
		True	False
Actual	True	663	337
	False	387	777

NaiveBayes

		Predicted	
		True	False
Actual	True	921	79
	False	657	507

RandomForests

		Predicted	
		True	False
Actual	True	866	134
	False	159	1005

MultilayerPerceptron

		Predicted	
		True	False
Actual	True	808	192
	False	231	933

Final Precision: 0.866
Final Recall: 0.873

Conclusions

Conclusions

- Our model is **highly accurate (88%)** at predicting confirmed exoplanet status
- Can be used as a tool to assist astronomers in confirming exoplanets
- Selecting priority follow-up observations
- Save valuable time: research telescopes, computing clusters, scientists
- No ML model has been developed for this dataset before
- Possible future direction: train a larger neural network

96% accuracy with astronomers’ input

Thanks for your attention ;)