Matilda Dingemans
Mr. Fisher
Computer Science III

Exoplanet Discovery and Analysis with AI Supervised Learning Methods

1) The Goal: The goal of my final project was to use data obtained from Nasa's Exoplanet archive from its 2009 Kepler Mission to predict whether, based on the statistics and information about the tracking and imagery of a star, the star contained any planet in its orbit, also known as "exoplanets" (a planet that orbits a star outside of our solar system). I used a Decision Tree classifier to determine whether or not a planet was an exoplanet and then also used Logistic Regression to calculate the probability of this determination. I also attempted to compare it to earth-like exoplanet in the dataset such as Kepler-1649 c[1] and Kepler-452b[2] using distance between points, to see whether or not other planets share similar attributes and if any research online can verify my findings.

2) The Kepler Mission:

The initial goal of the Kepler Mission was also to try to find other earth-like planets. The way the process worked was the Kepler space telescope (Launched in 2009) was used to record image data taken in very short intervals from thousands of bright stars in the Milky Way Galaxy. "When a planet passes in front of a star as viewed from Earth, the event is called a "transit". Transits by terrestrial planets produce a small change in a star's brightness of about 1/10,000. Kepler finds planets by looking for tiny dips in the brightness of a star when a planet crosses in front of it"[3]

For example: The tiny dot is a planet passing in front of a star, not a smudge on your computer screen. (This is an artist illustration)

"After a change in brightness is detected attributes of the planet can be calculated such as "the planet's orbital size can be calculated from the period (how long it takes the planet to orbit once around the star) and the mass of the star using Kepler's Third Law of planetary motion. The size of the planet is found from the depth of the transit (how much the brightness of the star drops) and the size of the star. From the orbital size and the temperature of the star, the planet's characteristic temperature can be calculated."[4]

[1] https://www.space.com/30172-six-most-earth-like-alien-planets.html
[2] https://www.planetary.org/articles/earth-like-worlds#:~:text=NASA%20 considers%20 exoplanet%20Kepler%2D452b,type%20star%20 similar%20to%20 ours
[3] Nasa, Kepler Mission Overview, https://www.nasa.gov/mission_pages/kepler/overview/index.html
[4] Nasa, Kepler Mission Overview.

3) Data Preparation

Now all of this information above as well as numerous other characteristics of each exoplanet is stored in Nasa's archives. The headings for columns of data that I used are below:
This is information about the planet:

- Name: A KOI name has an integer and a decimal part of the format KNNNNN.DD. The integer part designates the target star; the two-digit decimal part identifies a unique transiting object associated with that star.
- Disposition:The category of this KOI from the Exoplanet Archive. Current values are CANDIDATE, FALSE POSITIVE or CONFIRMED. This was the target value
- period: The interval between consecutive planetary transits.
- time0bk: The time corresponding to the center of the first detected transit in Barycentric Julian Day (BJD) minus a constant offset of 2,454,833.0 days.
- impact: The sky-projected distance between the center of the stellar disc and the center of the planet disc at conjunction, normalized by the stellar radius.
- duration: The duration of the observed transits. Duration is measured from first contact between the planet and star until last contact.
- depth: The fraction of stellar flux lost at the minimum of the planetary transit.
- prad: The radius of the planet. Planetary radius is the product of the planet star radius ratio and the stellar radius.
- teq:Approximation for the temperature of the planet. The calculation of equilibrium temperature assumes a) thermodynamic equilibrium between the incident stellar flux and the radiated heat from the planet
- insol: Insolation flux is another way to give the equilibrium temperature.
- Model_snr: Transit depth normalized by the mean uncertainty in the flux during the transits.

This is information about the star the planet orbits:

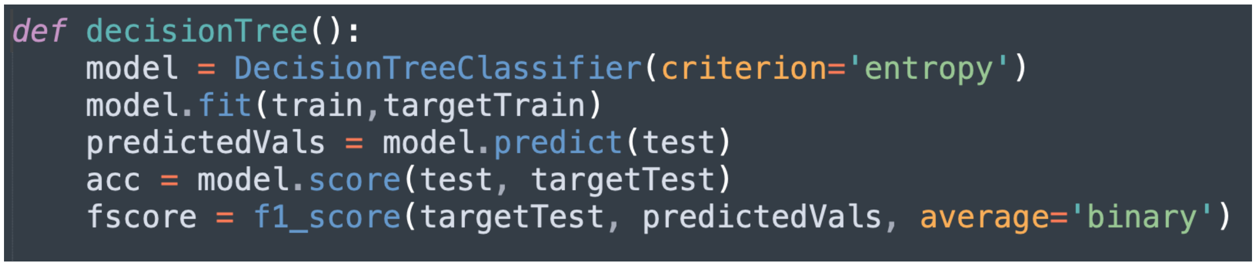
- steff: The photospheric temperature of the star.
- slogg: The base-10 logarithm of the acceleration due to gravity at the surface of the star.
- srad: The photospheric radius of the star
- ra: KIC (Kepler Input Catalog) Right Ascension
- dec: KIC Declination - unsure of what these are
- kepmag: Kepler-band (mag)

The first part of this project, as is with any supervised learning problem, was to clean and prepare the data. I first scanned the data and determined which columns I could remove (the data set I chose had more than double the amount of columns than what is stated above). I first removed the identifiers such as rowID or the name of the star, as well as removed any data, such as the score, that would be a dead giveaway to the answer of the prediction we were trying to make (I initially included score and since it is a value from 0-1, with any false positives being automatically set to 0, and I was surprised when my prediction returned a 98% correctness, don't worry, I removed it) Next, I removed four columns that were false positive flags. These columns were either 0 or 1, and if any of them were 1, the planet was automatically marked as a false positive. I obviously had to remove these, because, just like the score column, they would be a dead giveaway to what I was trying to predict. However, this did inspire a future idea (see section 7). Lastly, the data included margins of error, and I thought that these would not be that important to my predictions, however, I ran all of my functions on data with and without them and found that including the margins of error usually improved upon the prediction accuracy.

After I had decided which columns to remove, I then prepared the data by removing any rows with empty values, shuffling it, changing the target values from False Positive and Confirmed to 0 and 1, respectively, splitting the data, and standardizing all of the data (using skLearn's StandardScaler). I also noticed that while a majority of the rows were either confirmed or declared as a false positive, some of the rows (about 2000) were deemed as Candidates. I made a separate dataframe of all of these candidates before removing them from the core data set. I would then predict if these unconfirmed planets were exoplanets and then compare my predictions to a more recent dataset I found in Nasa's archive that has all of the same exoplanets, but some of the candidates have been either confirmed or declared as false positives.

4) Decision Tree predictions:

Now that all of my data had been prepped, it was time to actually start using some methods. At first, I incorrectly assumed that I could use Ridge Regression, but I later remembered that regression methods were used to predict values and were not used for binary problems, which was what my problem was. I quickly switched to using a decision tree classifier to predict the values. I also want to point out that skLearn has a function called "decisionTreeRegressor" which would not have been useful for my problem as that function is used to predict values, not classify them. Especially for my goal, I did not want to predict any numerical values, my data needed to predict binary classifications: 0 or 1. I used the skLearn model "DecisionTreeClassifier" to see how well the function would predict the test values.



```python
def decisionTree():
    model = DecisionTreeClassifier(criterion='entropy')
    model.fit(train,targetTrain)
    predictedVals = model.predict(test)
    acc = model.score(test, targetTest)
    fscore = f1_score(targetTest, predictedVals, average='binary')
```

	Entropy	Gini
Without Error Columns	Accuracy: 88% F1 Score: 82%	Accuracy: 88% F1 Score: 81%
With Error Columns	Accuracy: 90% F1 Score: 86%	Accuracy: 88% F1 Score: 85%

The best combination: With the error columns, and with the entropy criterion- I will use this combination for the rest of my analysis.

- The accuracy is the literal measure of how many it predicted right vs wrong
- The F1 score is the harmonic mean between precision and recall
    - We care more about maximizing the f1 score as it takes into account false positives and negatives, not just true positives and true negatives.
    - Good f1 score, 63% of data is false positives

To explore further, I wanted to understand which columns were actually affecting the outcomes the most. To do so, I used a skLearn function called "permutation_importance" which tells us the importance of the column when used to predict the classification. I then used the property called "importances_mean" on the results:
Koi_period: 0.125
Koi_time0bk: 0.0244
Koi_impact: 0.0536
Koi_duration: 0.0894
Koi_depth: 0.0639
Koi_prad: 0.1922
Koi_teq: 0.0938
Koi_insol: 0.0076
Koi_model_snr: 0.1847
Koi_steff: 0.0164
Koi_slogg: 0.0308
Koi_srad: 0.0349
Ra: 0.0386
Dec: 0.021
Koi_kepmag: 0.015

These values add up to 1, with a margin of error of about .05
*I also want to clarify that this was for one shuffle. The data importance will be variable since the data is shuffled differently each time. I also did not include the errors for this as I simply wanted to see which main attributes of the planets were most important.
From this list, we can see the most important variables are the

- Model_snr
    - (Transit depth (size of planet) normalized by the mean uncertainty in the flux (brightness) during the transits.)
- Period:
    - The interval between consecutive planetary transits - how long it takes to orbit
- Prad:
    - The radius of the planet.

After understanding which attributes of the data were post important, I wanted to test my model on the candidates data. Using the combination of entropy and using the error values (as that gives the highest fScore), I put the candidates through the model. I will not list all the values, but I will look at some specific predictions. I want to look at some specific planets that have been verified by a more recent data set posted by NASA (identified by Kepoi_Name).
From the older data (my data) to the new data:
K00799.01 went from candidate to false positive
K00804.01 went from candidate to confirmed
K00810.01 went from candidate to confirmed
K00841.03 went from candidate to confirmed
K00120.01 went from candidate to false positive

Now lets run the model on the candidate data and compare to the true values above.
0 = false positive
1 = Confirmed
K00799.01 = predicted 0 ✔
K00804.01 = predicted 1 ✔
K00810.01 = predicted 1 ✔
K00841.03 = predicted 1 ✔
K00120.01 = predicted 1 70% of the time ✖ - this was the only really variable one

I ran the code 10 times for each.

Since the data is shuffled differently each time, the last planet prediction changed every so often, but unfortunately, it mostly predicted the wrong value. All of the other candidates though were correct. ⅘ is not bad, and is a pretty good prediction rate. It matches up well with the score on the test data - around 80%

Obviously the model ran for all the candidates, but I have no way of verifying others. There are most likely other status updates in the new Dataset, but I feel that is essentially doing the same thing as using testing data where we know the status, so I will not be exploring more from the updated dataset.

5) Probability predictions
After I used a Decision Tree model to predict either 0 or 1, I wanted to see the actual probabilities of a planet being an exoplanet, and then look specifically at the planet above (K00120.01) in the candidates list and see the probability of it not being a exoplanet. To do so, I used skLearns ``Logistic Regression" function to spit out the probabilities and then evaluate it using log loss. In order to minimize the log loss function, I used a for loop to determine which C value would be best (to minimize log loss) and then used that value when making the model.

With the error bounds: Log Loss = ~22%
Without error bounds: Log Loss = ~35%

Safe to say the error bounds help lower the log loss.

I then printed out three items, the decision_function, the predicted probabilities of 0 and 1 (adding up to 1) and the actual values.

Here are some examples from 6 different Planets

Decision_function	Probability of False Positive	Probability of Confirmed	Actual Value
-0.04217638	0.51054253	0.48945747	FP
2.26004367	0.09448663	0.90551337	Confirmed
-14.8684857	0.99999965	0.00000035	FP
-55.46039359	1	0	FP
-17.60797787	0.99999998	0.00000002	FP
0.85052868	0.29932197	0.70067803	Confirmed

From this data - we can understand what the decision function actually does. When the data is predicted to be very clearly a False Positive, it is very negative, if the data is clearly confirmed, it is a positive number. If the model predicted exactly a 50/50 split, the decision function would return 0. The more sure the model is, the more extreme the value gets.

Lets inspect the output for planet K00120.01:
Running it 3 separate times:

Decision_function	Probability of False Positive	Probability of Confirmed	Actual Value
-0.400292690	0.59875798	0.40124202	False Positive
-0.76185967	0.68175735	0.31824265	False Positive
-0.614195832	0.64889733	0.35110267	False Positive

This outcome was surprising to me, because the model believed (correctly) that the planet should be labeled as a false positive, while the decision tree incorrectly labeled it. Even though the probabilities were only around 65% for False positives, it was still the majority in the correct prediction.

6) Finding possible habitable planets:
What I did next was compare all of the planets to planets that have been listed as earth-like. (based on this list). I used SciPy's euclidean distance function to find planets that had the most similar attributes.
Comparing Planets, I found that the most similar planets were: (running it three times)

Comparing to (Planets listed as earth like)	1	2	3
Kepler-452b	K08142.01 -	K00123.01 - Kepler-109 b	K06032.01 -
Kepler-1649 c	K02007.01- Kepler-1050 b	K03138.01 - Kepler-1649 b	K07386.01
Kepler-22 b	K03138.01 - Kepler-1649 b	K06011.01	K01009.01
Kepler- 69 c	K00139.01 - Kepler-111 c	K06924.01	K01697.01

The predictions without names (kepler-xyz) were deemed as false positives, but I still included them.
Using this data - we can see two things:

A) The Planets that are earth-like are (well obviously) similar - as two of them had the same planet predicted as most similar!
B) Kepler - 1050b , Kepler-109 b, and Kepler-111c, and Kepler-1649 b, all share similar attributes to earth! Very cool.

7) Predicting which False Positive identifiers were true:
My last inquiry into this data set was to determine if I would be able to predict whether or not a false positive identifier flag would be set off by setting that as my target value. Again, its binary. The flag is 1 when it's marked (making the whole thing a false positive), and 0 when it passes the test). I wanted to see which values were most important in predicting these false positives (again, using the permuation_importance function). I am going to again use a DecisionTree

Let's first understand what the false positive flags actually are (there are 4):
False positives can occur when:
1) the KOI is in reality an eclipsing binary star - koi_fpflag_nt
2) the Kepler light curve is contaminated by a background eclipsing binary - koi_fpflag_ss
3) stellar variability is confused for coherent planetary transits - koi_fpflag_co
4) instrumental artifacts are confused for coherent planetary transits - koi_fpflag_ec

Now, one by one, let's make these flags the target values and do some analysis.

- Koi_fpflag_nt
  - Most Important Prediction Features:
    - Koi_period - 0.1788
    - Koi_model_snr: 0.1634
    - Koi_duration: 0.1326
  - Accuracy: 0.8474697077690663
  - FScore: 0.6537216828478964
- Koi_fpflag_ss
  - Most Important Prediction Features:
    - Koi_prad: 0.2143
    - Koi_depth: 0.1443
    - Koi_period: 0.1279
  - Accuracy: 0.8367783321454028
  - FScore: 0.7424071991001123
- Koi_fpflag_co
  - Most Important Prediction Features:
    - Koi_period: 0.1999
    - Koi_impact: 0.1525
    - Koi_depth: 0.1991
  - Accuracy: 0.80256593014967
  - FScore: 0.6241519674355495
- Koi_fpflag_ec
  - Most Important Prediction Features:
    - Koi_period: 0.2148
    - Koi_duration: 0.1551
    - Koi_model_snr: 0.0743
  - Accuracy: 0.8282252316464719
  - FScore: 0.4904862579281184

The most important factor when flagging planets is the period as it appears in all of the functions. However, while it still appears for Koi_fpflag_ss, the most important factor is koi_prad, making up 20% of the decision. Prad is the radius of the star and this makes sense because the flag is marked when the light curve is contaminated - altering the size to a certain extent. The F Score is low for these because a majority of the values will be 0, with only a few being 1. This is because it includes not only the confirmed planets but also appears 0 when a different flag is sensed. From the fScores, the most common flag is most likely Koi_fpflag_ss as it has the highest fScore with Koi_fpflag_ec showing up the least.

8) Concluding remarks

Overall, this project was very interesting and I loved completing it, however I use that word softly, because there are infinite amounts of stuff to do with this data: it is never going to be complete. Throughout this project, especially early on, I ran into some difficulties simply because I had difficulty understanding some of the aspects of what I was trying to accomplish. I was using the wrong types of supervised learning methods or I was going about a problem the wrong way, however this project solidified my understanding of machine learning in a very effective way because it helped me understand different methods and why they worked as well as helped me dive into other aspects of machine learning from earlier in the year in different contexts. Another difficulty I found was always having precise data with it being shuffled. I did not want to seed the data because that might skew it, so if I were to add to this project, I would try to run each model on a bunch of different shuffles and average the values. I did this for some aspects of the code, and for every time I ran something, I ran it at least 3 times to see how different the outcomes were. If anything significantly changed, I could not make conclusions on the outcome, and I would not include it here in this write up. If I were to continue this project, I would look further into earth-like discovery as well as understanding the actual science behind this mission more. While I understand the basics, I would love to understand more about the specifics. For example, why do the different features mark specific flags more often? I have the data for which ones are most important, but being able to understand the astronomy and science behind why these are more likely is something I barely scratched the surface of during this project. All in all, this project was very fun and a great learning experience, thank you for a great year!
-Matilda

Sources:
https://www.nasa.gov/kepler/overview/abouttransits
https://scikit-learn.org/stable/modules/generated/sklearn.inspection.permutation_importance.html#sklearn.inspection.permutation_importance
https://www.space.com/39185-most-intriguing-alien-planets-2017.html
https://www.nasa.gov/mission_pages/kepler/overview/index.html
https://exoplanetarchive.ipac.caltech.edu/docs/API_kepcandidate_columns.html
https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.euclidean.html#scipy.spatial.distance.euclidean
Mr. Fishers Slides