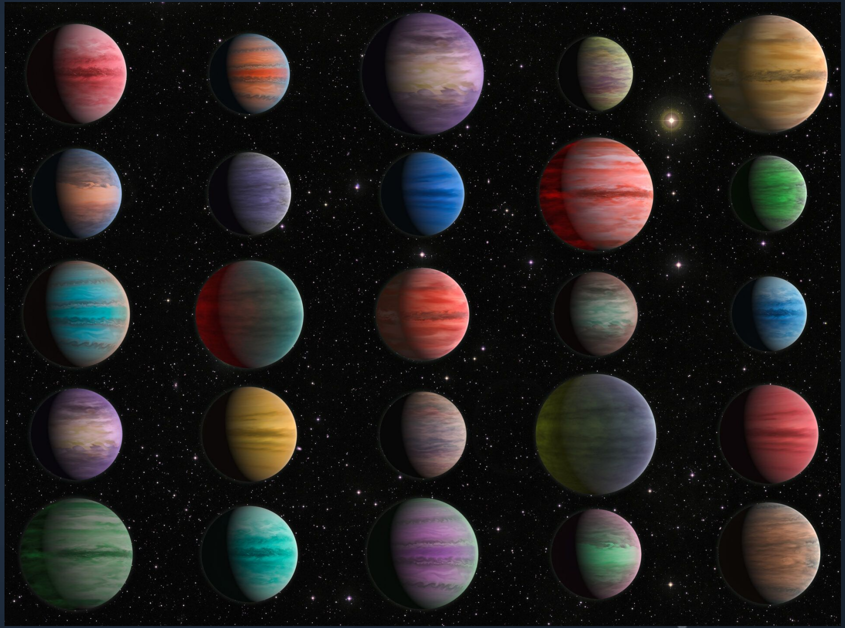
Planet Hunters

Instructor: Michel Lacerda
Group: Alvyn Kwon, Varun Gopal, Roneet Dhar, Ramana Manivannan, Zach Miller, Jasina Yu

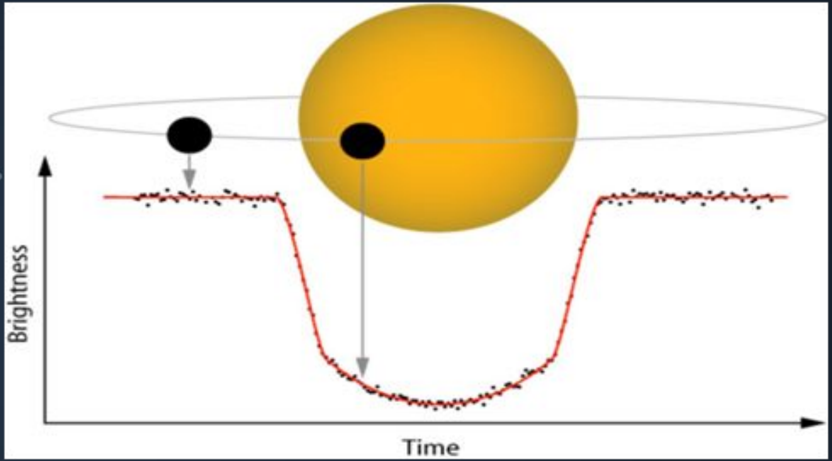
Project overview

- **Determining the existence of Exoplanets in our universe**
- Method known as Transit Photometry, used by astronomers/astrophysicists
- **Purpose:** Using Machine Learning/Visualizing Light Curves with given data to determine which model is best suited to find out the existence of exoplanets



Transit Photometry cont.

- Indirect method to view exoplanets
- Measure light flux from stars
- Determining exoplanet existence
  - Requires consistent patterns of flux dips
    - Dips are caused by planets orbiting in front
  - Limitation: exoplanets with very long orbit times may offer little evidence



NOTEBOOK I

UNDERSTANDING TRANSIT PHOTOMETRY & DETECTING EXOPLANETS

NOTEBOOK I

Data Used: NASA Dataset from the Kepler Space Telescope [First 5 Rows of the Data Frame]

	LABEL	FLUX.1	FLUX.2	FLUX.3	FLUX.4	FLUX.5	FLUX.6	FLUX.7	FLUX.8	FLUX.9	...	FLUX.3188	FLUX.3189	FLUX.3190	FLUX.3191	FLUX.3192	FLUX.3193	FLUX.3194	FLUX.3195	FLUX.3196	FLUX.3197
0	1	93.85	83.81	20.10	-26.98	-39.56	-124.71	-135.18	-96.27	-79.89	...	-78.07	-102.15	-102.15	25.13	48.57	92.54	39.32	61.42	5.08	-39.54
1	1	-38.88	-33.83	-58.54	-40.09	-79.31	-72.81	-86.55	-85.33	-83.97	...	-3.28	-32.21	-32.21	-24.89	-4.86	0.76	-11.70	6.46	16.00	19.93
2	1	532.64	535.92	513.73	496.92	456.45	466.00	464.50	486.39	436.56	...	-71.69	13.31	13.31	-29.89	-20.88	5.06	-11.80	-28.91	-70.02	-96.67
3	1	326.52	347.39	302.35	298.13	317.74	312.70	322.33	311.31	312.42	...	5.71	-3.73	-3.73	30.05	20.03	-12.67	-8.77	-17.31	-17.35	13.98
4	1	-1107.21	-1112.59	-1118.95	-1095.10	-1057.55	-1034.48	-998.34	-1022.71	-989.57	...	-594.37	-401.66	-401.66	-357.24	-443.76	-438.54	-399.71	-384.65	-411.79	-510.54

- 3197 Flux Columns ⟶Represent the brightness of a star at regular intervals

Separating Exoplanets from Non-Exoplanets

- Label 1: Star has an Exoplanet
- Label 0: False Positive Source/Not an Exoplanet

```
labels = flux_data.LABEL
flux_data = flux_data.drop('LABEL',axis=1)
non_exo_data=flux_data.loc[labels==0]
exo_data=flux_data.loc[labels==1]
```

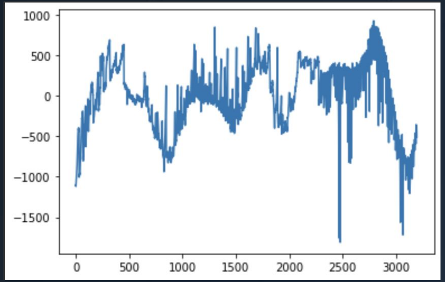
Training Data: Exoplanets vs Non-Exoplanets

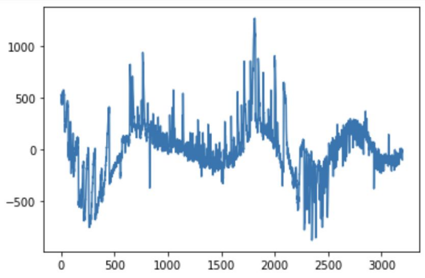
```
print ("Number of exoplanets:", len(exo_data))
print ("Number of non-exoplanets:", len(non_exo_data))
```

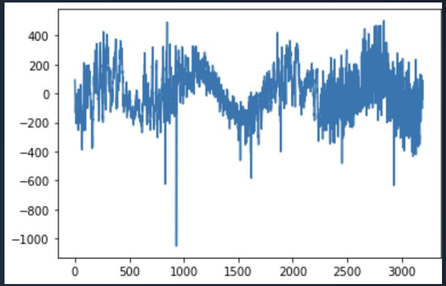
```
Number of exoplanets: 37
Number of non-exoplanets: 5050
```

VISUALIZING LIGHT CURVES

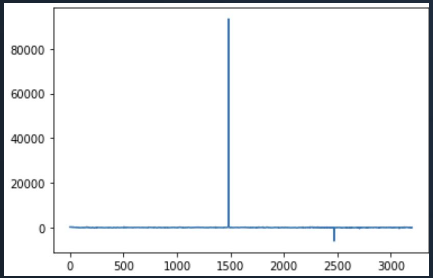
```
for i in range(3):
  print('Exoplanet Light Curve '+ str(i))
  plot_light_curve(exo_data, i)
```

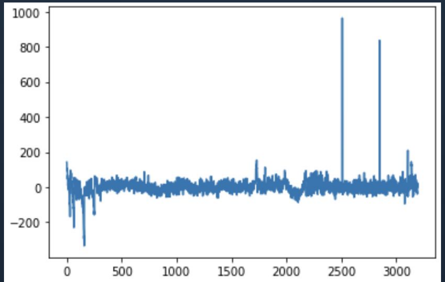


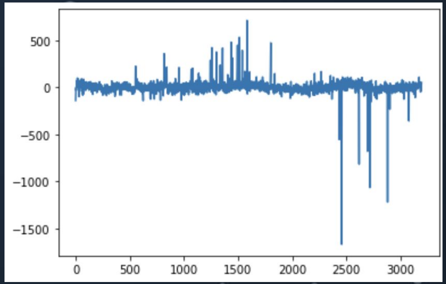




```
for i in range(3):
  print('Non-Exoplanet Light Curve '+ str(i))
  plot_light_curve(non_exo_data, i)
```



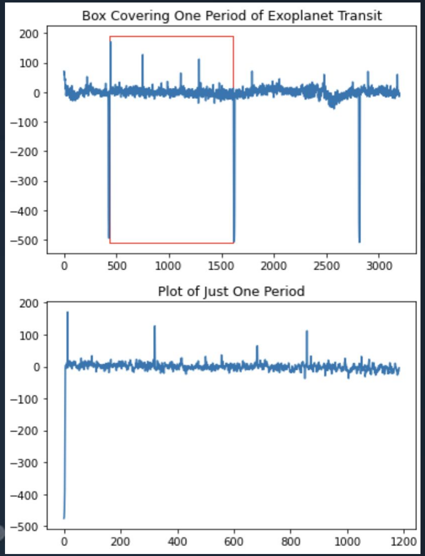




LIGHT CURVE FOLDING

- When determining if a light curve is an exoplanet or not, one of the crucial components is determining the planet's period

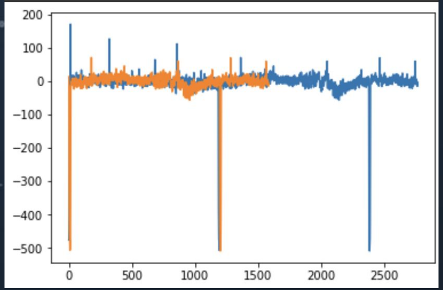
```
index = 12 #@param {type:"slider", min:0, max:37, step:1}
t_0 = 430 #@param {type:"slider", min:0, max:3197, step:1}
period = 1184 #@param {type:"slider", min:0, max:3197, step:1}

from matplotlib.patches import Rectangle
light_curve=np.array(exo_data.loc[index])
plt.plot(light_curve)
plt.title('Box Covering One Period of Exoplanet Transit')
plt.gca().add_patch(Rectangle((t_0, -510), period, 700, linewidth=1,edgecolor='r',facecolor='none'))
plt.show()

plt.plot(light_curve[t_0: t_0+period])
plt.title('Plot of Just One Period')
plt.show()
```

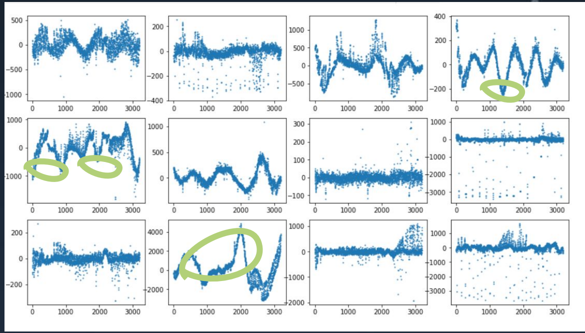


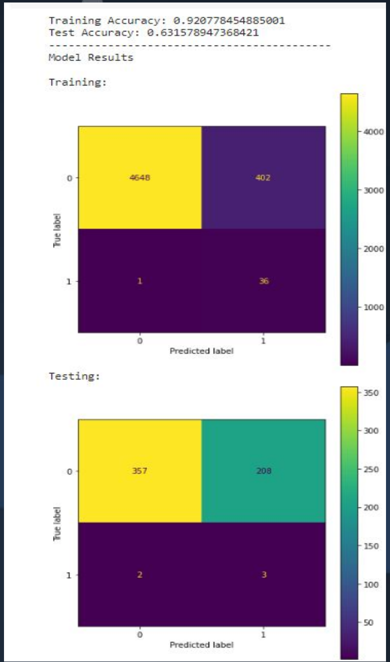
- Visualizes the period from the first dip (t_0)
- Period Length (Dip to Dip)

- Light Curve Folding: Determines if the period is the same for all dips
- Folding plots all the periods on top of each to see if there is a consistent trend

```
start_period_1= t_0 # time of first transit
plt.plot(light_curve[start_period_1:]) # plots the first curve in blue

#Plot the curve starting from Period 2
start_period_2= t_0 + period # time of first transit
plt.plot(light_curve[start_period_2:]) # plots the first curve in blue
```

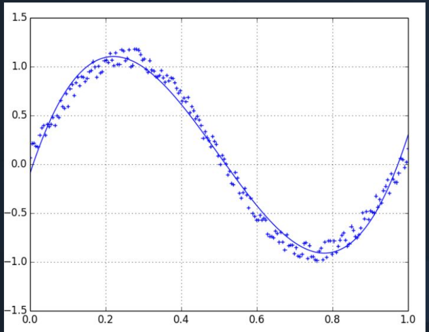


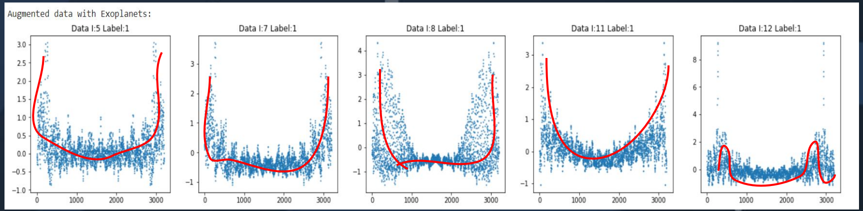




NOTEBOOK 2

ANALYZING VARIOUS AI CLASSIFICATION MODELS TO DETECT EXOPLANETS





BASIC OVERVIEW

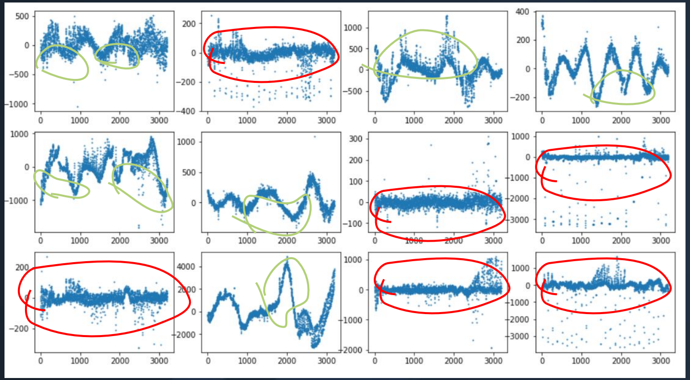
- Primary Objective - Test Different Classification Models to Detect Exoplanets

- Tested Models
  - KNN Clustering
  - Logistic Regression
  - Decision Trees
- Key focus
  - Data augmentation

- Critical Areas to Consider
  - Defining True Accuracy
  - Impact of Imbalanced Datasets
  - More Complex Methods

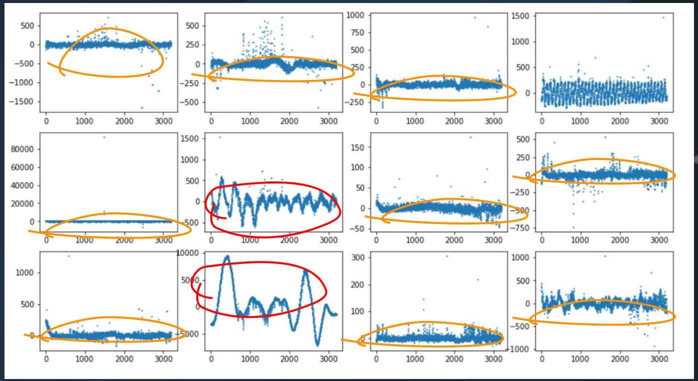
Flux Curves

EXOPLANETS



- Patterns
  - Regular dips at consistent intervals (green)
  - Sine wave shape with multitude of varying degree

NONEXOPLANETS



- Patterns
  - Mostly fairly constant flux measures (orange)
- **Problem to Consider**
  - Several graphs show similarity with exoplanet data (red)

KNN ALGORITHM

- Relies on Clustering to Separate exoplanet and non-exoplanet data
- Nearest Neighbors = 5 (increases accuracy)

```python
#@title Run this to load helper functions and create train_X, train_y, test_X, t

def analyze_results(model, train_X, train_y, test_X, test_y):
    """
    Helper function to help interpret and model performance.

    Args:
    model: estimator instance
    train_X: {array-like, sparse matrix} of shape (n_samples, n_features)
    Input values for model training.
    train_y : array-like of shape (n_samples,)
    Target values for model training.
    test_X: {array-like, sparse matrix} of shape (n_samples, n_features)
    Input values for model testing.
    test_y : array-like of shape (n_samples,)
    Target values for model testing.

    Returns:
    None
    """
    print("-----------------------------------------")
    print("Model Results")
    print("")
    print("Training:")
    fig = plt.figure(figsize=(22,7))
    ax = fig.add_subplot(1,3,1)
    plot_confusion_matrix(model,train_X,train_y,ax=ax,values_format = '.0f')
    plt.show()
    print("Testing:")
    fig = plt.figure(figsize=(22,7))
    ax = fig.add_subplot(1,3,1)
    plot_confusion_matrix(model,test_X,test_y,ax=ax,values_format = '.0f')
    plt.show()

def reset(train,test):
    train_X = train.drop('LABEL', axis=1)
    train_y = train['LABEL'].values
    test_X = test.drop('LABEL', axis=1)
    test_y = test['LABEL'].values
    return train_X,train_y,test_X,test_y

train_X,train_y,test_X,test_y = reset(df_train, df_test)
```

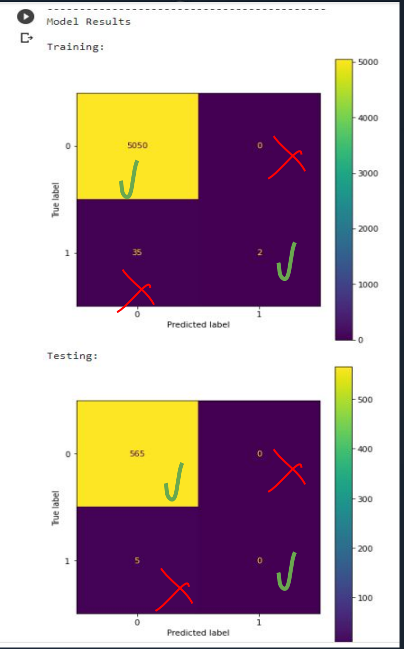
```python
n_neighbors = 5

model = KNeighborsClassifier(n_neighbors)
```

```python
model.fit(train X, train y)
```

KNN RESULTS

```
# Calculate the predictions and accuracies on train_X and test_X
# using our trained model

train_predictions = model.predict(train_X)
test_predictions = model.predict(test_X)
#YOUR CODE HERE
train_accuracy = accuracy_score(train_y, train_predictions)
test_accuracy = accuracy_score(test_y, test_predictions)

print("Train Accuracy: " + str(train_accuracy))
print("Test Accuracy: " + str(test_accuracy))

####################

Train Accuracy: 0.9931197169254964
Test Accuracy: 0.9912280701754386
```

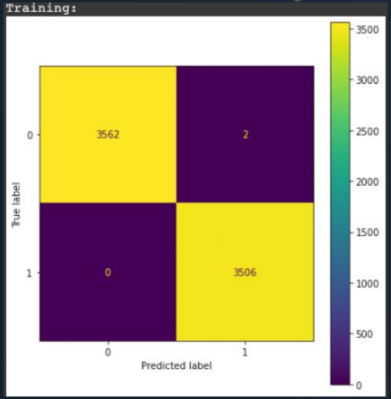
- Accuracy: ~99.31% (train) , ~99.12% (test)
  - What is the true accuracy?

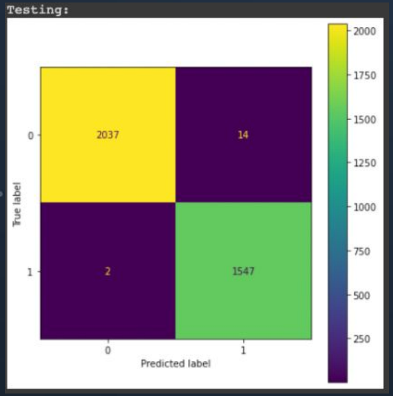


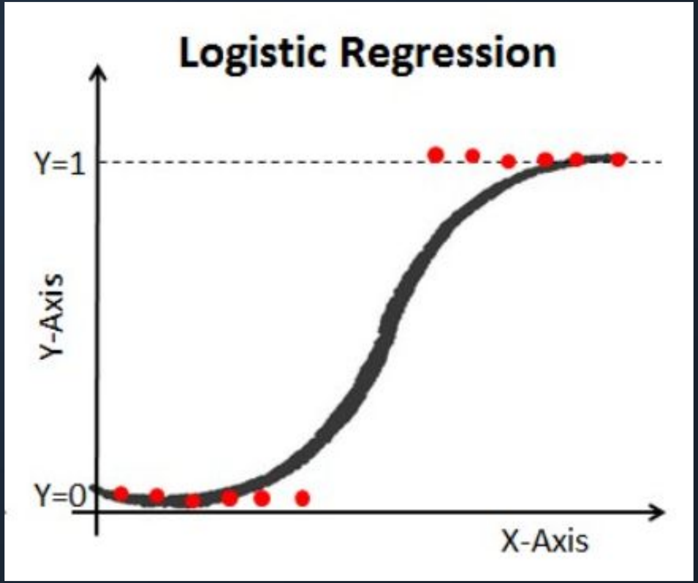
- Confusion Matrix Findings
  - Dearth of exoplanets
  - Several false positives - (0, 1) or (1, 0)

LOGISTIC REGRESSION

- Assigns each datapoint a probability of being an exoplanet. Decides based on this probability whether or not datapoint is an exoplanet.
- Max Iterations: 1000.
  - Sets how many times model runs to learn.
- Accuracy: ~99.97% (train) , ~99.56% (test)
- Confusion Matrix Findings
  - No big flaws.
  - Most common mistake is to incorrectly identify a planet as an exoplanet.

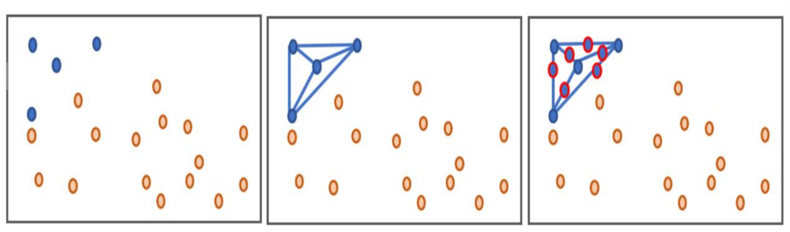


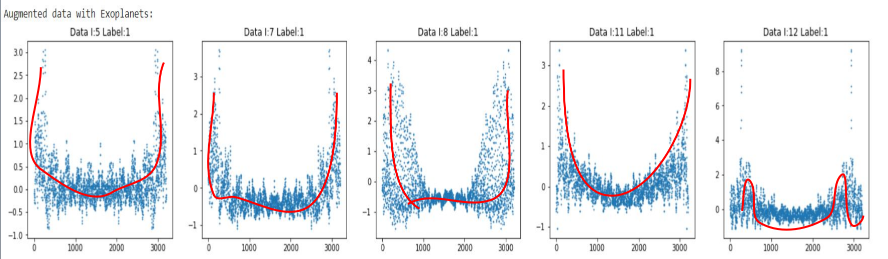




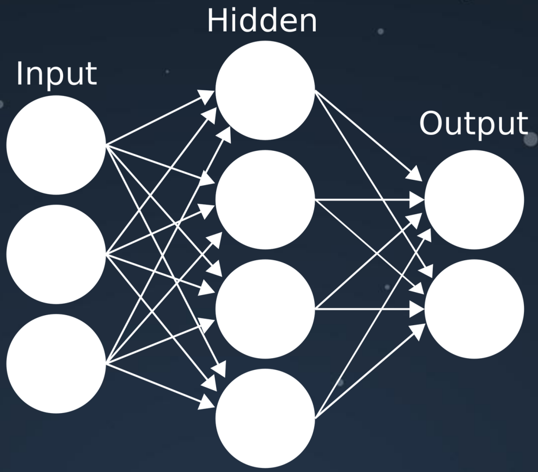
DATA AUGMENTATION

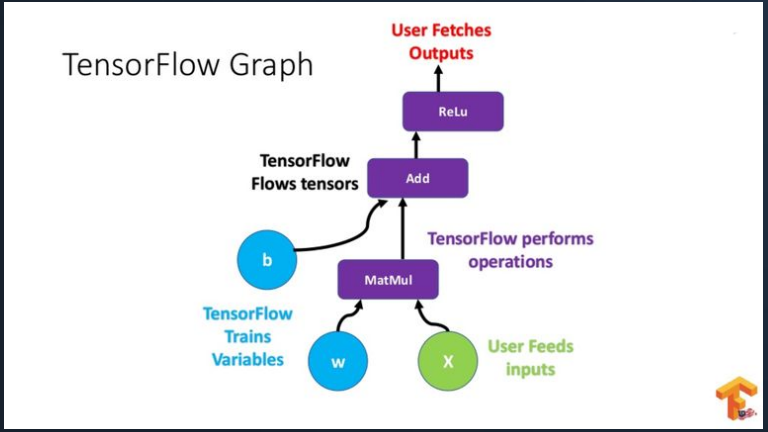
- We used SMOTE (Synthetic Minority Oversampling Technique) to augment our data.
- Using existing minority data (exoplanets), SMOTE creates **more minority data to help balance the dataset**. This ensures that the model will not have a bias towards the majority data.

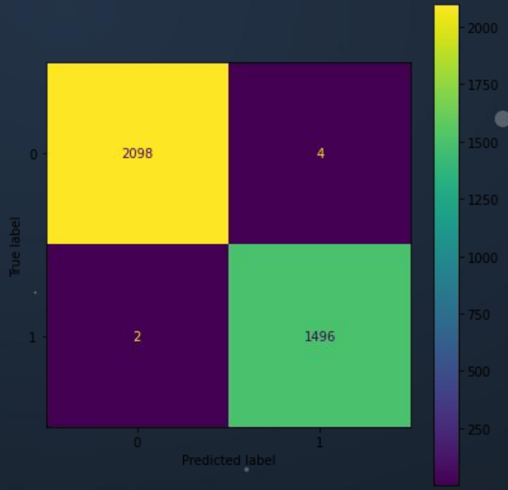




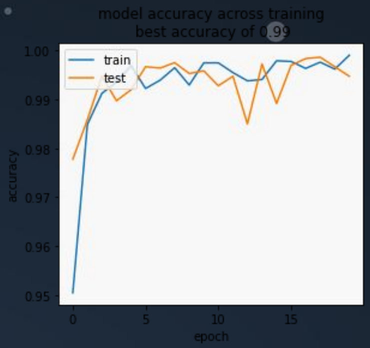
NOTEBOOK 3
COMPLEX AND MODERN ALGORITHMS



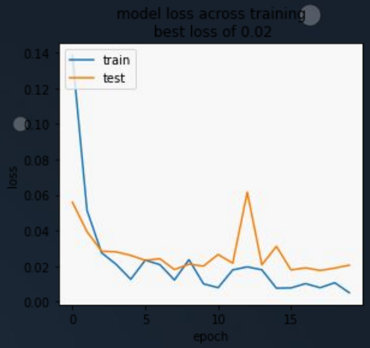




NOTEBOOK 3



Accuracy



loss

Complex Machine Learning Algorithms

- MLP (Multi-layer Perceptron) with augmented data
  - Provided an accuracy of 99.83%
  - Confusion Matrix displayed on for 1496 exoplanets being identified only two were misclassified
- Neural Networks (Tensorflow and Keras)
  - Yielded an accuracy 99.47%

Neural Network

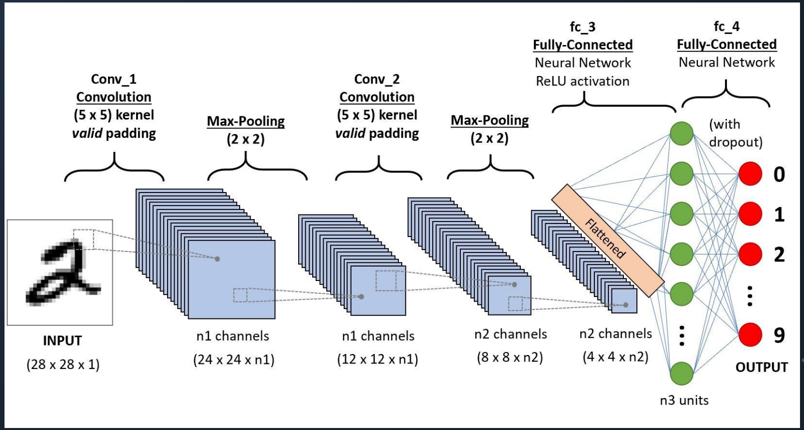
```
# YOUR CODE HERE to train the model

batch_size = 64
epochs = 20
validation_data = (aug_test_X, aug_test_y)
verbose = 1
shuffle = True

history = model.fit(aug_train_X, aug_train_y, batch_size=batch_size, epochs=epochs, verbose=verbose, validation_data=validation_data, shuffle=shuffle)
#####################
```

NOTEBOOK 3 CONT.

Complex Machine Learning Algorithms

- CNN (convolutional Neural network)
  - Accuracy yielded is 99.67%
  - Best at not missing exoplanets

```
[ ]
# Train the Model

#######TODO#########

#parameter stuff
batch_size = 64
epochs = 20
validation_data = (cnn_aug_test_X, cnn_aug_test_y)
verbose = 1
shuffle = True

#training
history = model.fit(cnn_aug_train_X, cnn_aug_train_y, batch_size=batch_size, epochs=epochs, verbose=verbose,
                    validation_data=validation_data, shuffle=shuffle)

####################
```



Final Consideration / AI MODEL

What what your best AI model?

- Logistic Regression Using Augmented Data

How high was the performance?

- 99.6% During Testing

How would you interpret this performance?

- Confusion Matrix

What steps did you take to improve the model performance?

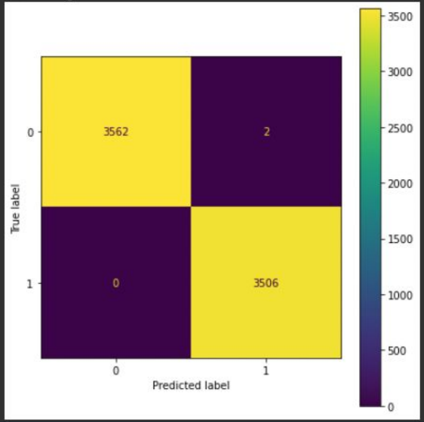
- Augmentation (Before and After Augmentation)

Where did your model fail?

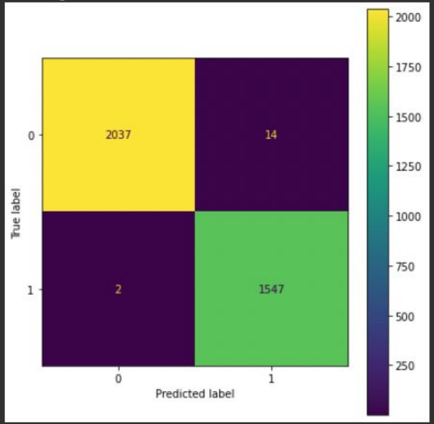
- 99.6% is not enough. A larger dataset would be better.

```
model = LogisticRegression(max_iter=1000)
model.fit(aug_train_X, aug_train_y)
train_predictions = model.predict(aug_train_X)
test_predictions = model.predict(aug_test_X)
print(accuracy_score(aug_train_y, train_predictions))
print(accuracy_score(aug_test_y, test_predictions))
analyze_results(model=model, train_X=aug_train_X, train_y=aug_train_y, test_X=aug_test_X, test_y=aug_test_y)
```

```
0.9997171145685997
0.9955555555555555
--------------------------------------------
Model Results

Training:
```



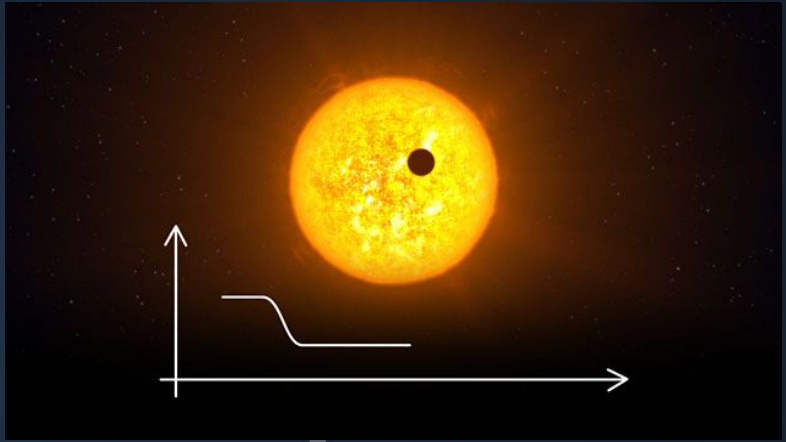
Testing:



Real-World Application

How might you operationalize this in the real world?

- **NASA's efforts currently use the transit photometry method** as a reliable and logically provable way to measure the consistency of orbits and therefore the presence of an exoplanet.
- **The use of the logistic regression model provides the easiest, most robust, and reliable way** or method of accessing and conveying information relevant to transit photometry, specifically when considering factors such as ease of use, readability, and relative simplicity to code.



What more data would you collect?

- An additional aspect of data that would greatly help the entire process would be to **use transit photometry as a means to gather more data about the exoplanet itself**, once the patterns of dips have been determined and the exoplanet been confirmed. Qualities of the exoplanet like mass could be easily expanded upon, whereas more complicated details like orbiting moons, atmospheric data, and type of planet could be collected using a potentially more powerful descendant or enhancement of transit photometry, especially when considering the possibilities that could be collected using present methods.