Aalto University

CS-C3240 Machine Learning

Using Machine Learning to predict the existence of exoplanets

Project Report Part 1

Santeri Paljakka
santeri.paljakka@aalto.fi

Mikko Alikärri
mikko.alikarri@aalto.fi

Returned: October 13, 2023

Contents

1 Introduction **3**

2 Problem formulation **3**

3 Methods **3**
3.1 Data preparation 3
3.2 Logistic regression 4
3.3 Deep learning 5
3.4 Validation 5

4 Results **5**

5 Conclusion **6**

A Code **6**

B Confusion matrices **19**

C Logistic loss for train and validation sets **19**

1 Introduction

Space beyond our solar system has evoked curiosity for as long as we have known about its existence. Finding planets orbiting a star other than ours is the first step in the search for life outside the Earth. To shed light on these intriguing questions, NASA uses advanced data analysis and machine learning on data collected with, e.g., space telescopes [2].

This project uses logistic regression and deep learning methods to classify possible observations of exoplanets to false positives and data points containing information about an actual exoplanet. Section 2 introduces the data and problem statement in more detail. Section 3 presents the used machine learning methods. Section 4 displays the key results, which are then concluded in Section 5. The Python code used is given as an appendix.

2 Problem formulation

The data points used in this project represent observations of a star hosting one or more transiting planets gathered in the Kepler mission by Nasa using a space telescope. These observations are called Kepler Objects of Interest (KOI). Each data point contains measurements from the star as well as information on whether an exoplanet was confirmed to exist at the target location. In this project, the existence of exoplanets is predicted using the measurement data provided by Nasa and logistic regression. The data is found on the website of the Nasa Exoplanet Science Institute which the California Institute of Technology operates [3]. We are using the table Q1-Q17 DR24 Done which is constructed by Thomson et al. (2018) [4].

The task is a supervised learning task, and we use the variable Disposition Using Kepler Data as our label. The label is presented by a categorical variable 1 if a candidate for an exoplanet is found, and 0, if no exoplanet is found and the observation is marked as false positive. The categorization process is described by Thomson et al. (2018) [4].

The Kepler exoplanet data has been used for machine learning projects before e.g. in a project of [1]. However, our project uses different selection of features, preprocessing, and machine learning methods.

3 Methods

3.1 Data preparation

First, the data is cleaned and more descriptive names are assigned to the columns. The raw data from the Nasa database contains 49 columns, including names, indexes, information about methodology, error terms, and indicators from previous data analysis. This irrelevant information was left out, on top of terms that clearly depend on other measurements, when selecting the features used in the study. We also tested, if the numerical values would be statistically significant for the ML method accuracy, but anything significant did not raise up. The selected 13 features describe actual measurements by the Kepler space telescope and are listed below:

- OrbitalPeriod.[Days] - The interval between consecutive planetary transits
- ImpactParameter - 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

- TransitDuration.[Hours] - The duration of the observed transits
- TransitDepth.[ppm] - The fraction of stellar flux lost at the minimum of the planetary transit
- PlanetaryRadius - The radius of the planet
- EquilibriumTemperature.[K] - Approximation for the temperature of the planet
- InsolationFlux - Insolation flux is another way to give the equilibrium temperature
- TransitSignalToNoise - Transit depth normalized by the mean uncertainty in the flux during the transits
- StellarSurfaceGravity - The base-10 logarithm of the acceleration due to gravity at the surface of the star
- RA.[deg] - Exoplanets position coordinate
- Dec.[deg] - Exoplanets position coordinate
- KeplerMagnitude.[mag] - Exoplanets magnitude to Kepler

Explanations are fetched from Nasas Data Columns in Kepler Objects of Interest Table.

Altogether, when the null values are cleaned, the data set contains 7374 data points. Each data point represents a possible observation of an exoplanet. The feature data is normalized using StandardScaler from sklearn prerocessing library.

3.2 Logistic regression

We use logistic regression to classify the data points into binary categories. Logistic regression is widely used and efficient for binary classification problems like this because of its ability to separate the output into ones and zeros.

With logistic regression, we use polynomial hypothesis space as it enables better fit when there is no certainty of linear dependency with the label. Visualization is difficult with 13 features but the best possible polynomial degree is attained by iterating over all possible degrees from 1 to 4. In polynomial classification, the algorithm computes the predicted labels by using a polynomial map of the features. In our case, the decision boundary is a polynomial function and decision regions are half-spaces as the used labels are binary.

The quality of the decision boundary is evaluated using the logistic loss function, which is calculated as

$$(y, y') = -1/m \sum_{i=1}^{m} [y \cdot \log(y') + (1 - y) \cdot \log(1 - y')].$$

The logistic loss function takes in two label vectors, the true label and the predicted probability estimate for a predicted label. The predicted label is calculated with the ML method's predict method and the probability estimate is calculated with predict-proba method. The logistic loss is readily implemented in sklearn library and thus a logical choice for us.

3.3 Deep learning

For the second machine learning model, we chose deep learning because of its hypothesis space's (artificial neural network) potential to assign different weights to various parameters with multiple layers. Deep learning is a supervised learning method that, for training itself, uses the provided features and compares the results with the label vector. In our project, we chose to train multilayer perception (MLP), where the data moves from one layer to another, to solve our classification problem.

To understand the hypothesis map of our project, it's essential to grasp the structure of the multi-layer neural network we employ. This network comprises numerous layers (e.g., layers 1, 2, 3, ...), and each layer is composed of multiple cells. When a feature input is supplied to the deep learning model, the information is distributed with unique weights to all the cells within layer one. Then all the cells send out the modified data to the second label with unique weights added, and so on. Furthermore, each cell is associated with a unique bias that influences the input value of that cell. Bias is present in all the labels but the last. Our neural network had earlier defined 13 feature inputs, consisting of 36 layers and 15 cells per layer.

With the multilayer perception classification model, we naturally chose the logistic loss function for its usability in the supervised classification problem. In the code, the logistic loss function is fetched from sklearn.metrics library and the equation can be seen in the earlier section. In addition, the number of layers was chosen to minimize the cumulative logistic loss function value. In the code, this was done with a for-loop where we tried different layer numbers of 1,4,8,12,20,28,36,44,56, and 72

3.4 Validation

Data is validated using a separate testing set. To avoid deterministic separations of data, we use random sampling when dividing the data into training, validation, and testing sets. The data set size is relatively big, and therefore a smaller proportion of the data for testing is sufficient. We put 75 % of data into a training set, 12,5 % into validation, and the remaining 12,5 % into a testing set. The separate validation set is used to attain the optimal polynomial degree for the polynomial classification. There is no correlation between different data points as the data points describe different planets and are thus independent.

4 Results

In this section, we will go through the results obtained from the implemented machine learning methods. We will compare the errors obtained from different sets of data (Table 1), training, validation, and test sets (see section 3.4 Validation). In addition, there is an accuracy score for the test set.

The errors were calculated with the logistic loss function represented above. Logistic regression gave somewhat worse values than the deep learning method.

Table 1: Errors calculated with logistic loss and accuracy score for the test set given to three significant figures.

ML method	Training error	Validation error	Test error	Accuracy score for test set
Logistic regression	0.437	0.510	0.634	81.2%
Deep learning	0.442	0.462	0.462	81.0%

We also implemented confusion matrices for both of the ML methods. (Figure 1 in Appendix B) The distribution of the matrices differs significantly. In logistic regression, the amount of false positive samples was increased compared to the deep learning method, where the amounts of false positive and false negative samples were approximately the same.

Overfitting was prevented using a separate validation set and iterating over both polynomial degree in the case of logistic regression and number of layers in the deep learning method. By minimizing the validation error, polynomial degree of 2 and neural network depth of 36 were chosen. The validation and training errors are visualized in figure 2 Appendix C .

5 Conclusion

This section will conclude this report, which aims to validate the existence of exoplanets using logistic regression and deep learning. Finding exoplanets is the first step in the search for life outside our solar system. We use data from the Kepler mission by Nasa to categorize our data points into confirmed planets and false positives. First, the data is preprocessed and 13 features describing the measurements by the space telescope is chosen for our model. We use logistic regression with polynomial hypothesis space and artificial neural networks to predict our label. Both of these methods are powerful in binary classification problems and do not require much assumptions on the data distribution. Separate training, validation and test sets are used to train the model, to optimize the model parameters and to assess the final results respectively.

Neither of our models succeed to predict the existence of planets perfectly as the accuracy in both cases is around 80 %. In terms of accuracy, the logistic regression model seems to perform slightly better with the accuracy of 81,2 %. However, the deep learning model has lower error especially with the validation and test sets. Also, the number of false positives and negatives is more balanced with the deep learning method. It can be concluded that the results still leave room for improvement in the future research.

The complexity of the data set and the scientific knowledge needed to understand the full scope of the problem set challenges in the model and feature selection. In the future, possibilities with other methods could be explored. In the article of Araujo (2020), even accuracy of around 95 % was achieved using a random forest method [1].

A Code

The code has been done with Python into a ipynb file format. The libraries used can be seen in the first code section.

Machine Learning project

- Validate exoplanets Kepler disposition with machine learning methods

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import log_loss, accuracy_score,
confusion_matrix, f1_score, classification_report
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, PolynomialFeatures

from sklearn.neural_network import MLPClassifier
```

Dataset

```
df = pd.read_csv('q1_q17_dr24_koi_2023.09.19_04.32.41.csv')
# Rename the columns into more readable names
df.columns
=['KepID',"KOIName","KeplerName","ExoplanetArchiveDisposition",
"ExoplanetDispositionKepler","DispositionScore","NotTransit-
LikeFalsePositiveFlag",
"koi_fpflag_ss","CentroidOffsetFalsePositiveFlag","EphemerisMatchIndic
atesContaminationFalsePositiveFlag",
"OrbitalPeriod.[Days]","OrbitalPeriodErr1.[Days]","OrbitalPeriodErr2.
[Days]",
"TransitEpoch-bk","TransitEpoch-bkErr1","TransitEpoch-bkErr2",
"ImpactParameter","ImpactParameterErr1","ImpactParameterErr2",
"TransitDuration.[Hours]","TransitDurationErr1.
[Hours]","TransitDurationErr2.[Hours]",
"TransitDepth.[ppm]","TransitDepthErr1.[ppm]","TransitDepthErr2.
[ppm]",
"PlanetaryRadius","PlanetaryRadiusErr1","PlanetaryRadiusErr2",
"EquilibriumTemperature.[K]","EquilibriumTemperatureErr1.
[K]","EquilibriumTemperatureErr2.[K]",
"InsolationFlux","InsolationFluxErr1","InsolationFlux2",
"TransitSignalToNoise",
"TCEPlanetNumber","TCEDeliveryName",
"StellarEffectiveTemperature.[K]","StellarEffectiveTemperatureErr1.
[K]","StellarEffectiveTemperatureErr2.[K]",
"StellarSurfaceGravity","StellarSurfaceGravityErr1","StellarSurfaceGra
vityErr2",
"StellarRadius","StellarRadiusErr1","StellarRadiusErr2",
"RA.[deg]","Dec.[deg]","KeplerMagnitude.[mag]"]
```

```
print("Amount of the columns in the dataset", len(df.columns))
```

```
Amount of the columns in the dataset 49
```

Modify dataframe

Modify and exoplanets disposition calculated by Kepler into binary form. This will be our label vector.

```
df['ExoplanetCandidate'] =
df['ExoplanetDispositionKepler'].apply(lambda x: 0 if x == 'FALSE
POSITIVE' else 1)
```

Clean the dataframe

```
"""
Selected columns:
"OrbitalPeriod.[Days]",
"TransitEpoch-bk",
"ImpactParameter",
"TransitDuration.[Hours]",
"TransitDepth.[ppm]",
"PlanetaryRadius"
"EquilibriumTemperature.[K]",
"InsolationFlux",
"TransitSignalToNoise",
"StellarSurfaceGravity",
"RA.[deg]","Dec.[deg]","KeplerMagnitude.[mag]"]"""

df =
df.drop(columns=['KepID',"KOIName","KeplerName","ExoplanetArchiveDispo
sition",
"ExoplanetDispositionKepler","DispositionScore","TransitEpoch-
bk","NotTransit-LikeFalsePositiveFlag",
"koi_fpflag_ss","CentroidOffsetFalsePositiveFlag","EphemerisMatchIndic
atesContaminationFalsePositiveFlag",
"OrbitalPeriodErr1.[Days]","OrbitalPeriodErr2.[Days]",
"TransitEpoch-bkErr1","TransitEpoch-bkErr2",
"ImpactParameterErr1","ImpactParameterErr2",
"TransitDurationErr1.[Hours]","TransitDurationErr2.[Hours]",
"TransitDepthErr1.[ppm]","TransitDepthErr2.[ppm]",
"PlanetaryRadiusErr1","PlanetaryRadiusErr2",
"EquilibriumTemperatureErr1.[K]","EquilibriumTemperatureErr2.[K]",
"InsolationFluxErr1","InsolationFlux2",
"TCEPlanetNumber","TCEDeliveryName",
"StellarEffectiveTemperature.[K]","StellarEffectiveTemperatureErr1.
```

```
[K]","StellarEffectiveTemperatureErr2.[K]",
"StellarSurfaceGravityErr1","StellarSurfaceGravityErr2",
"StellarRadius","StellarRadiusErr1","StellarRadiusErr2"])
df.head()
```

```
   OrbitalPeriod.[Days]  ImpactParameter  TransitDuration.[Hours]  \
0              2.470613           0.8186                  1.74259
1              2.204735           0.0010                  3.88216
2              4.887803           0.0260                  2.36386
3              3.849372           0.9193                  2.66050
4              4.780328           0.9516                  2.03490

   TransitDepth.[ppm]  PlanetaryRadius  EquilibriumTemperature.[K]  \
0             14186.0            12.85                      1344.0
1              6690.6            16.39                      2025.0
2              4342.1             4.84                       801.0
3              1317.3            13.10                      2035.0
4               977.2             7.07                      1396.0

   InsolationFlux  TransitSignalToNoise  StellarSurfaceGravity   RA.
[deg]  \
0          772.22                6802.0                  4.455
286.80847
1         3973.70                6714.5                  4.021
292.24728
2           97.10                2207.8                  4.590
297.70935
3         4055.29                 262.6                  3.657
294.35654
4          898.71                 383.4                  4.003
289.73972

   Dec.[deg]  KeplerMagnitude.[mag]  ExoplanetCandidate
0  49.316399                11.338                   1
1  47.969521                10.463                   1
2  48.080853                 9.174                   1
3  38.947380                11.432                   1
4  44.647419                11.665                   1
```

Drop rows from dataframe that include Nan value

```
df.isna().any()
print("Shape of the original data format", df.shape)
# Drop the rows that include Nan
df_cleaned = df.dropna()
print("Shape of the cleaned data format", df_cleaned.shape)
```

```
Shape of the original data format (7470, 13)
Shape of the cleaned data format (7374, 13)
```

Features and labels

Lets create 2 validation sets from the cleaned data frame

```
# Create a feature matrix X and label vector y.
y = df_cleaned['ExoplanetCandidate'].to_numpy()
X = df_cleaned.drop(columns=['ExoplanetCandidate']).to_numpy()
# Create test and training sets
    # First split the data into train set that contains 75 %
    # and test set that contains 25 % of the data
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25)
    # Second split the test set into test set that contains 50 % of
the test set
X_test2, X_val, y_test2, y_val = train_test_split(X_test, y_test,
test_size=0.5)

# Check the distribution of the labels
print("Number of zeros in ytrain:", np.sum(y_train == 0))
print("Number of ones in ytrain:", np.sum(y_train == 1))
# Same for second split
print("Number of zeros in ytest:", np.sum(y_test == 0))
print("Number of ones in ytest:", np.sum(y_test == 1))
```

```
Number of zeros in ytrain: 2315
Number of ones in ytrain: 3215
Number of zeros in ytest: 798
Number of ones in ytest: 1046
```

Features in feature matrix

```
for column in df_cleaned.drop(columns=['ExoplanetCandidate']).columns:
    print(column)
```

```
OrbitalPeriod.[Days]
ImpactParameter
TransitDuration.[Hours]
TransitDepth.[ppm]
PlanetaryRadius
EquilibriumTemperature.[K]
InsolationFlux
TransitSignalToNoise
StellarSurfaceGravity
RA.[deg]
Dec.[deg]
KeplerMagnitude.[mag]
```

Polynomial logistic regression model

Create a logistic regression model, find the best polynomial and train it with scaled data. Print the results.

```
save_regression = [1000]

clf_tr_error = []
clf_val_error = []
save_clf = []
polys = []
degrees = range(1,5)

for degree in degrees:
    # Create a polynomial logistic regression model
    clf_poly = LogisticRegression(random_state=10, C=100,
max_iter=10000)
    # Choose the degree of the polynomial
    poly = PolynomialFeatures(degree)
    # Create the polynomial train features
    X_train_poly = poly.fit_transform(X_train)
    # Scale the data with StandardScaler
    scaler = StandardScaler().fit(X_train_poly)
    X_train_poly = scaler.transform(X_train_poly)
    # Create the polynomial test features
    X_val_poly = poly.fit_transform(X_val)
    # Scale the data with StandardScaler
    scaler = StandardScaler().fit(X_val_poly)
    X_val_poly = scaler.transform(X_val_poly)
    # Train the model
    clf_poly.fit(X_train_poly, y_train)
    # Predict labels
    y_pred_poly = clf_poly.predict(X_val_poly)
    # Calculate probabilities on the validation set
    y_prob_val = clf_poly.predict_proba(X_val_poly)
    # Calculate logistic loss for the validation set
    val_error = log_loss(y_val, y_prob_val)

    # Calculate training error
    y_pred_train = clf_poly.predict(X_train_poly)
    y_prob_train = clf_poly.predict_proba(X_train_poly)
    tr_error = log_loss(y_train, y_prob_train)
    polys.append(poly)

    clf_tr_error.append(tr_error)
    clf_val_error.append(val_error)
    save_clf.append(clf_poly)

poly_degree = np.array(clf_val_error).argmin() + 1
```

```
clf_poly = save_clf[poly_degree - 1]
tr_error = clf_tr_error[poly_degree -1]
val_error = clf_val_error[poly_degree-1]
poly = polys[poly_degree-1]

# Print the results
print("Training error: ", tr_error)
print("Error for validation set: ", val_error)
print("Degree of the polynomial: ", poly_degree)
```

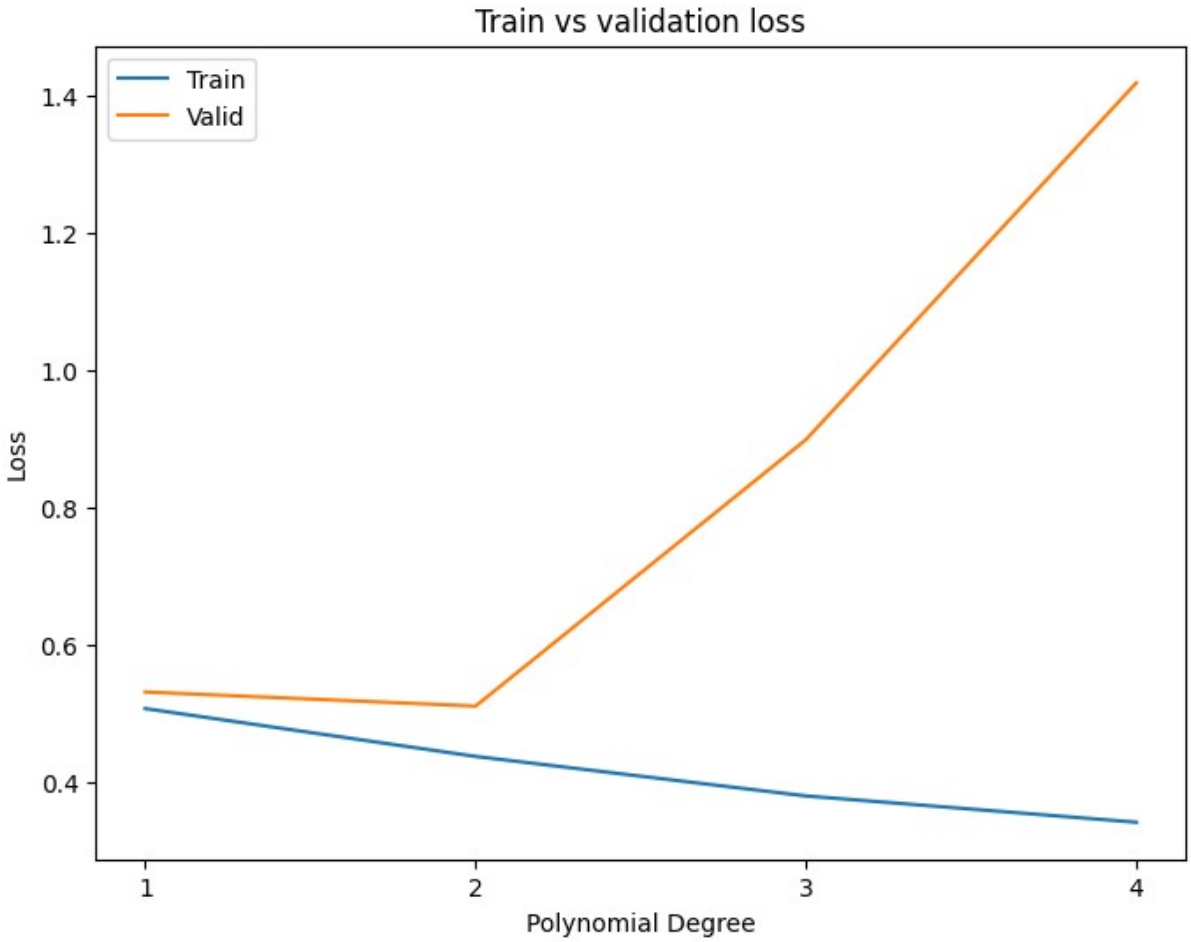
```
Training error:  0.4365577940040843
Error for validation set:  0.5098748938601099
Degree of the polynomial:  2
```

Test the trained model with other test set defined in up.

```
X_test_poly = poly.fit_transform(X_test2)
scaler = StandardScaler().fit(X_test_poly)
X_test_poly = scaler.transform(X_test_poly)

y_pred = clf_poly.predict(X_test_poly) # Predict labels
accuracy = accuracy_score(y_test2, y_pred)
y_prob_test = clf_poly.predict_proba(X_test_poly)
tr_error = log_loss(y_test2, y_prob_test)
print("Accuracy Score for test set: ", accuracy)
print("Test set error: ", tr_error)
```

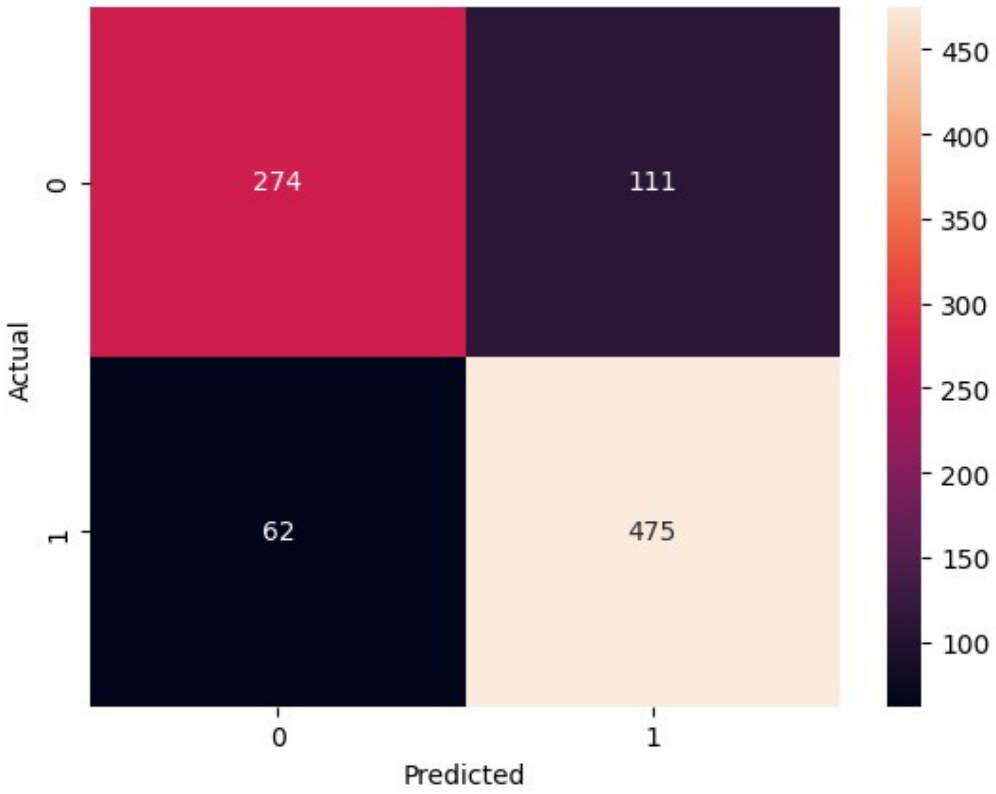
```
Accuracy Score for test set:  0.8123644251626898
Test set error:  0.6340723338550187
```

```
plt.figure(figsize=(8, 6))

plt.plot(degrees, clf_tr_error, label = 'Train')
plt.plot(degrees, clf_val_error,label = 'Valid')
plt.xticks(degrees)
plt.legend(loc = 'upper left')

plt.xlabel('Polynomial Degree')
plt.ylabel('Loss')
plt.title('Train vs validation loss')
plt.show()

errors = {"Polynomila degree":degrees,
          "clf_train_error":clf_tr_error,
          "clf_val_error":clf_val_error,
         }
pd.DataFrame(errors)
```



	Polynomila degree	clf_train_error	clf_val_error
0	1	0.506242	0.530426
1	2	0.436558	0.509875
2	3	0.378993	0.898013
3	4	0.340721	1.417772

```
# Plot the confusion matrix
confusionMatrix = confusion_matrix(y_test2, y_pred)
sns.heatmap(confusionMatrix, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```




```
# Classification report
print("Classification Report: \n", classification_report(y_test2,
y_pred))
```

```
Classification Report:
               precision    recall  f1-score   support

           0       0.82      0.71      0.76       385
           1       0.81      0.88      0.85       537

    accuracy                           0.81       922
   macro avg       0.81      0.80      0.80       922
weighted avg       0.81      0.81      0.81       922
```

Deep-learning model

Create and test deep-learning model with different layers

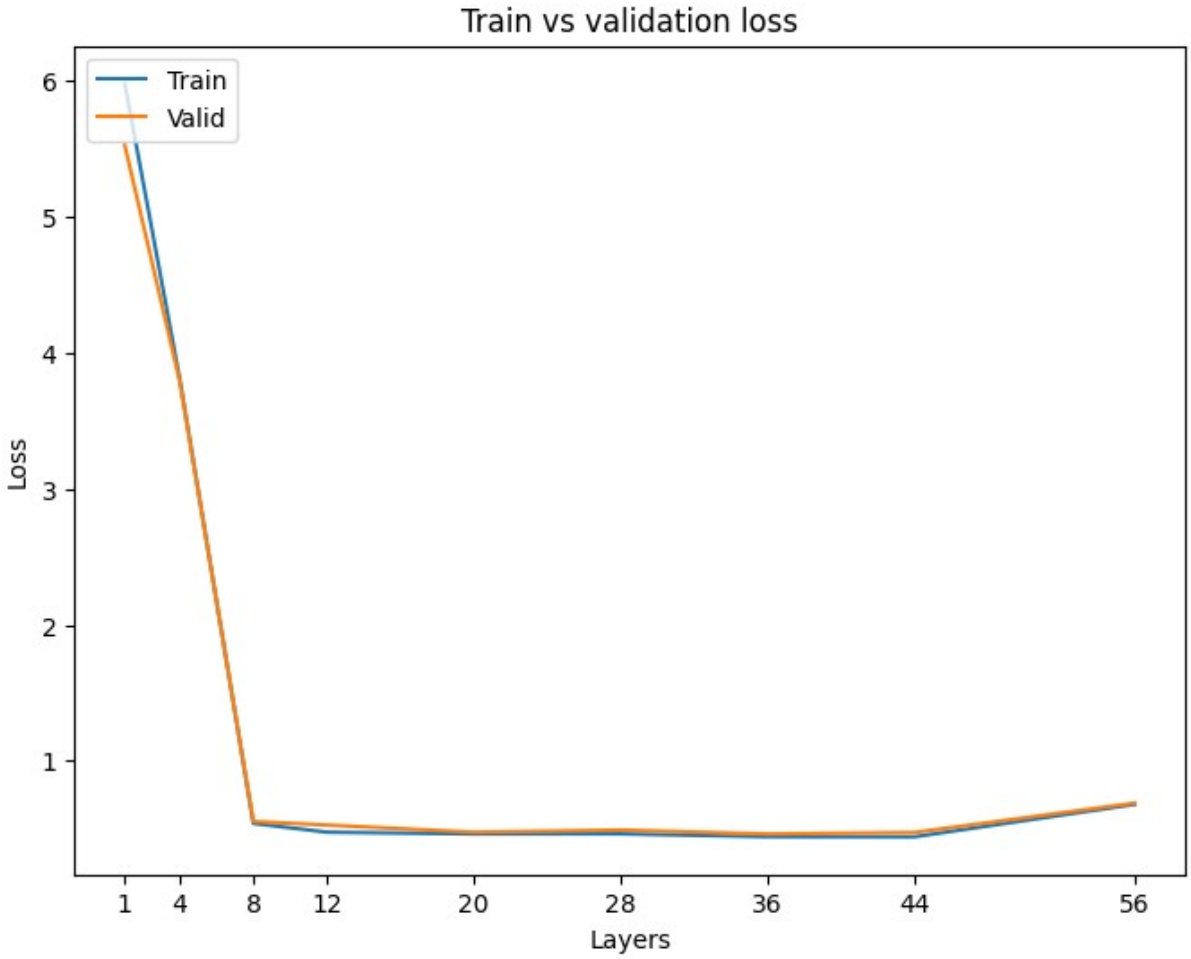
```
## define a list of values for the number of hidden layers
num_layers = [1,4,8,12,20,28,36,44,56]    # number of hidden
layers ,16,20,28,36,44,56,72
```

```
num_neurons = 15  # number of neurons in each layer

# we will use this variable to store the resulting training errors
corresponding to different hidden-layer numbers
mlp_tr_error = []
mlp_val_error = []
save_mlp = []

for i, num in enumerate(num_layers):
    hidden_layer_sizes = tuple([num_neurons]*num) # size (num of
neurons) of each layer stacked in a tuple

    mlp_regr = MLPClassifier(max_iter=20000, hidden_layer_sizes =
hidden_layer_sizes)
    mlp_regr.fit(X_train, y_train)

    ## evaluate the trained MLP on both training set and validation
set
    y_pred_train = mlp_regr.predict(X_train)    # predict on the
training set
    y_prob_train = mlp_regr.predict_proba(X_train)
    tr_error = log_loss(y_train, y_prob_train)    # calculate the
training error
    y_pred_val = mlp_regr.predict(X_val) # predict values for the
validation data
    y_prob_val = mlp_regr.predict_proba(X_val)
    val_error = log_loss(y_val, y_prob_val) # calculate the validation
error

    mlp_tr_error.append(tr_error)
    mlp_val_error.append(val_error)
    save_mlp.append(mlp_regr)

final_index = np.array(mlp_val_error).argmin()
number_of_layers = num_layers[final_index]

mlp_regr = save_mlp[final_index]

print('Áfter validation, the number of layers is: ', number_of_layers)
```

```
Áfter validation, the number of layers is:  36
```

```
plt.figure(figsize=(8, 6))

plt.plot(num_layers, mlp_tr_error, label = 'Train')
plt.plot(num_layers, mlp_val_error,label = 'Valid')
plt.xticks(num_layers)
plt.legend(loc = 'upper left')

plt.xlabel('Layers')
```

```
plt.ylabel('Loss')
plt.title('Train vs validation loss')
plt.show()

errors = {"num_hidden_layers":num_layers,
          "mlp_train_error":mlp_tr_error,
          "mlp_val_error":mlp_val_error,
         }
pd.DataFrame(errors)
```



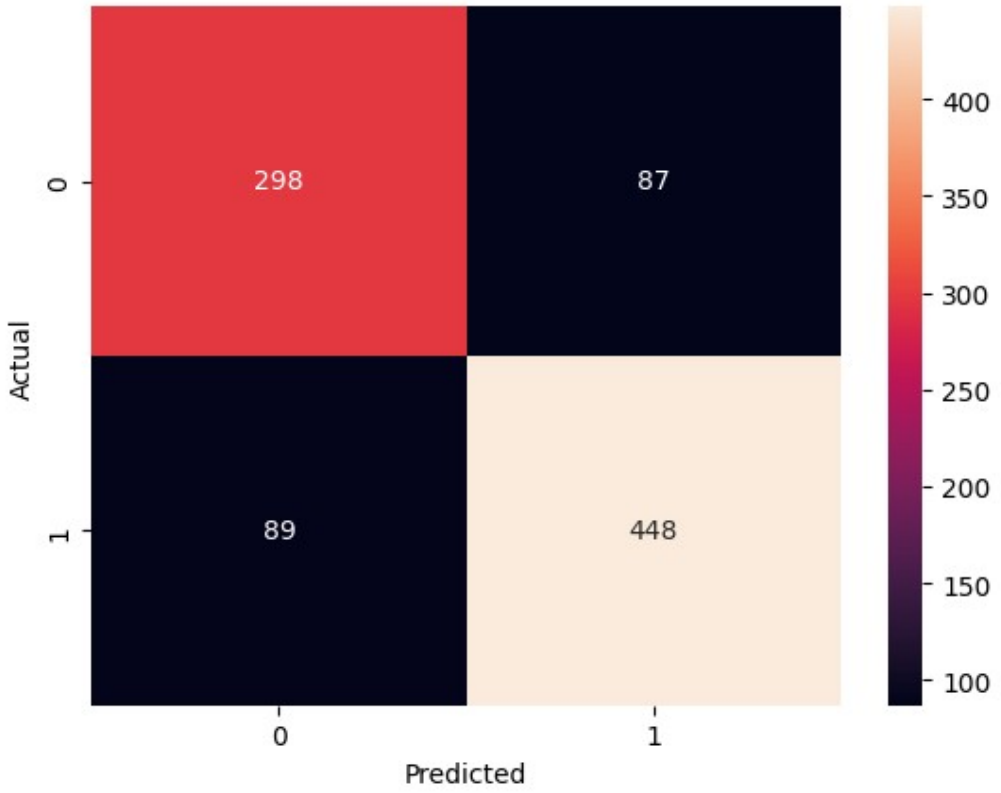
	num_hidden_layers	mlp_train_error	mlp_val_error
0	1	5.981515	5.527445
1	4	3.803425	3.786486
2	8	0.539801	0.553938
3	12	0.475193	0.527754
4	20	0.462389	0.474835
5	28	0.462940	0.491030
6	36	0.441669	0.461680

```
7                  44          0.441291       0.473111
8                  56          0.679866       0.689100
```

```
y_pred_deepLearning = mlp_regr.predict(X_test2) # Predict labels
accuracy = accuracy_score(y_test2, y_pred_deepLearning)
y_prob_test = mlp_regr.predict_proba(X_val)
test_error = log_loss(y_val, y_prob_test) # calculate the validation
error
print("Accuracy Score for test set: ", accuracy)
print("Test error", test_error)
```

```
Accuracy Score for test set:  0.8091106290672451
Test error 0.46168006293690067
```

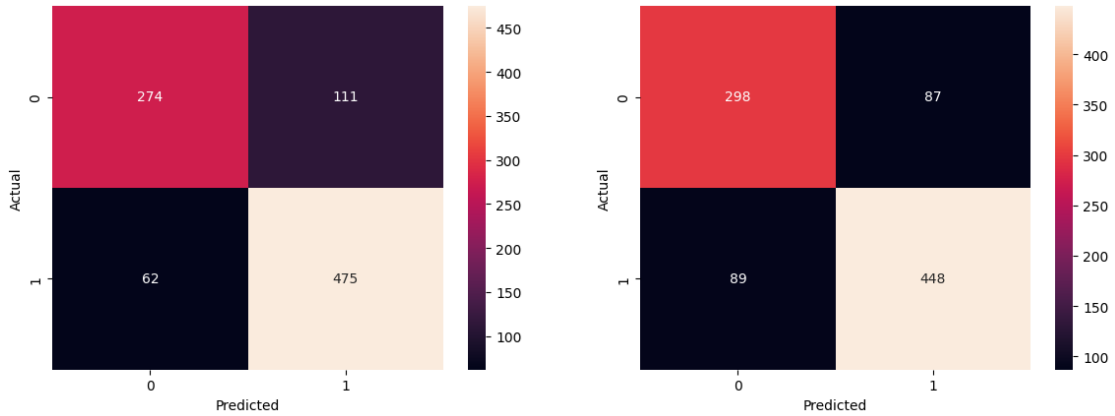
```
# Plot the confusion matrix
confusionMatrix = confusion_matrix(y_test2, y_pred_deepLearning)
sns.heatmap(confusionMatrix, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```




```
# Classification report
print("Classification Report: \n", classification_report(y_test2,
y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.71      0.76       385
           1       0.81      0.88      0.85       537

    accuracy                           0.81       922
   macro avg       0.81      0.80      0.80       922
weighted avg       0.81      0.81      0.81       922
```

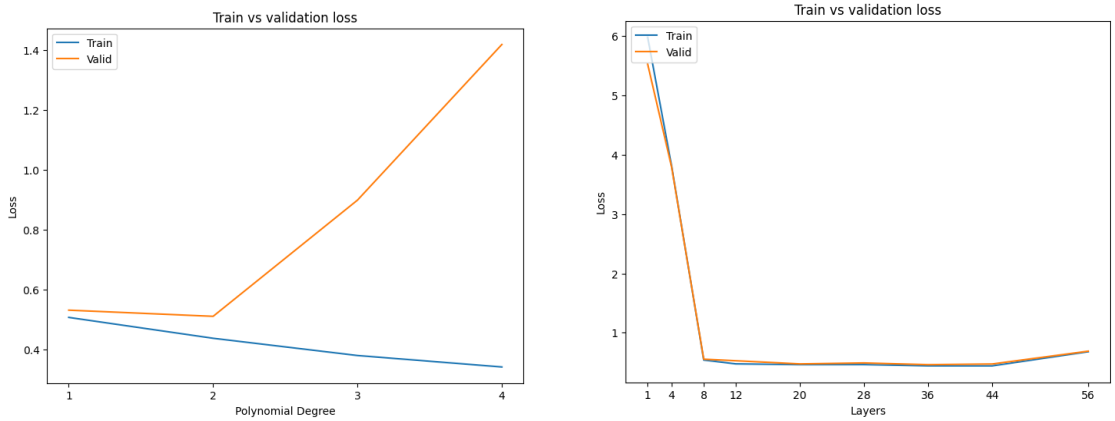
B Confusion matrices



(a) Logistic Regression (b) Deep Learning

Figure 1: The confusion matrices of both models

C Logistic loss for train and validation sets



(a) Logistic Regression (b) Deep Learning

Figure 2: Logistic loss error for train and validation sets

References

[1] Ismael Araujo. Using machine learning to find exoplanets with nasa's data. *Towards Data Science*, 2020.

[2] Rachel Hoover. New deep learning method adds 301 planets to kepler's total count. *https://www.nasa.gov/feature/ames/new-deep-learning-method-adds-301-planets-to-keplers-total-count*, 2021.

[3] Nasa. Kepler objects of interest (koi) activity table. 2017. Edited: 31.8.2017. Accessed: 20.9.2023. Available from: `https://exoplanetarchive.ipac.caltech.edu/cgi-bin/TblView/nph-tblView?app=ExoTbls&config=koi`.

[4] Susan E. Thompson, Jeffrey L. Coughlin, Kelsey Hoffman, Fergal Mullally, Jessie L. Christiansen, Christopher J. Burke, Steve Bryson, Natalie Batalha, Michael R. Haas, Joseph Catanzarite, Jason F. Rowe, Geert Barentsen, Douglas A. Caldwell, Bruce D. Clarke, Jon M. Jenkins, Jie Li, David W. Latham, Jack J. Lissauer, Savita Mathur, Robert L. Morris, Shawn E. Seader, Jeffrey C. Smith, Todd C. Klaus, Joseph D. Twicken, Jeffrey E. Van Cleve, Bill Wohler, Rachel Akeson, David R. Ciardi, William D. Cochran, Christopher E. Henze, Steve B. Howell, Daniel Huber, Andrej Prša, Solange V. Ramírez, Timothy D. Morton, Thomas Barclay, Jennifer R. Campbell, William J. Chaplin, David Charbonneau, Jørgen Christensen-Dalsgaard, Jessie L. Dotson, Laurance Doyle, Edward W. Dunham, Andrea K. Dupree, Eric B. Ford, John C. Geary, Forrest R. Girouard, Howard Isaacson, Hans Kjeldsen, Elisa V. Quintana, Darin Ragozzine, Megan Shabram, Avi Shporer, Victor Silva Aguirre, Jason H. Steffen, Martin Still, Peter Tenenbaum, William F. Welsh, Angie Wolfgang, Khadeejah A. Zamudio, David G. Koch, and William J. Borucki. Planetary Candidates Observed by Kepler. VIII. A Fully Automated Catalog with Measured Completeness and Reliability Based on Data Release 25. , 235(2):38, April 2018. doi: 10.3847/1538-4365/aab4f9.