ECON 441B : Intro Machine Learning Lab

Week 1, Lecture 1 | Github and Google Colab

Sam Borghese

Wednesday, January 9th, 2024

1. Course Overview
2. Github
3. Google Colab
4. In-Class Assignment

Course Overview

Econ 441B : ~~Applied Data Management For Economists~~

Econ 441B : Introductory Machine Learning Lab

Course Description:

This is an introductory lab to Machine Learning for Economists. The content will be taught through visualization with real data and pragmatic in class assignments with real or fabricated data. The tasks of the assignments will match the most popular fields for Quantitative Economists mainly focussing on quantitative financial analytics and consumer analytics. The main goal of the course is to arm students with a set of critical thinking skills and algorithms to solve any data problem using a machine learning approach.

The course will synchronize with Econ 425 and provide hands-on examples for the concepts learned inside of that course. The implications of the examples in 441B will also be useful in the Asset Pricing course which will take the tools learned in Machine Learning and apply them to Quantitative Finance. While the ML and Asset Pricing programs synergize they can be taken independently.

This course was introduced to give an idea of how the algorithms are used outside of the textbook and academia

Econ 441B : Introductory Machine Learning Lab

This course aligns with Professor Grigory's 425 and Professor Guang's 425

All courses are independent, but I assume you attended their weekly lectures before my course

This course counts for 20% of 425 Grade
441B Grade = 425 Grade

No curve in this class but maybe in 425

Econ 441B : Introductory Machine Learning Lab

Topics

You learn the topics in their class. Practice them in my class

Topics:

Week 1: Introduction & Machine Learning Pipeline
Week 2: Discrete Classification I - Logistic Regression
Week 3: Linear Models, Regularization, and Hyperparameter Tuning
Week 4: Discrete Classification II - Decision Trees
Week 5: Dealing with Imbalanced Data
Week 6: Neural Nets
Week 7: Large-Language Models
Week 8: LLM Cont. & Bagging and Boosting (Non-Parametric Models)
Week 9: Unsupervised Learning - Clustering & PCA
Week 10: Reinforcement Learning

Econ 441B : Introductory Machine Learning Lab

Assignments

Computation of Course Grade:

- 8 - 12.5% In-Class Assignments (There are 10 total, you can miss 2 of them)
- All work must be posted to GitHub to count
- This grade will count towards 20% of 425 grade
- The grade you get in 425 will be your 441B grade as well

There will be no curve. In-class assignments graded for correctness, with partial credit given.

Each class you will be given a short challenge problem that you will have 10-30 minutes to complete

Assignments should be completed in class but you will have until Friday to Submit
No Late Work Accepted

Econ 441B : Introductory Machine Learning Lab

Guest Presentations

Twice there will be presentations from industry people who use Machine Learning in their jobs commonly. They will share what algorithms, softwares and packages they use to solve various problems.

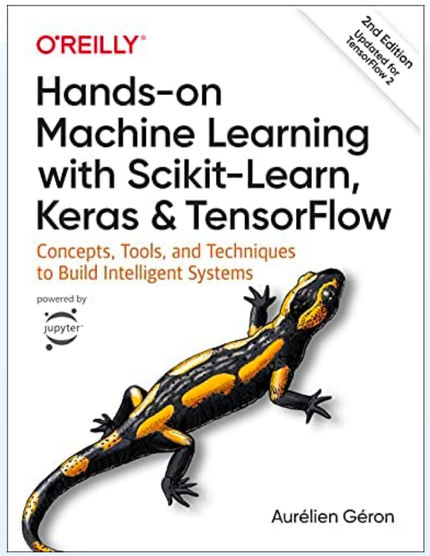
Econ 441B : Introductory Machine Learning Lab

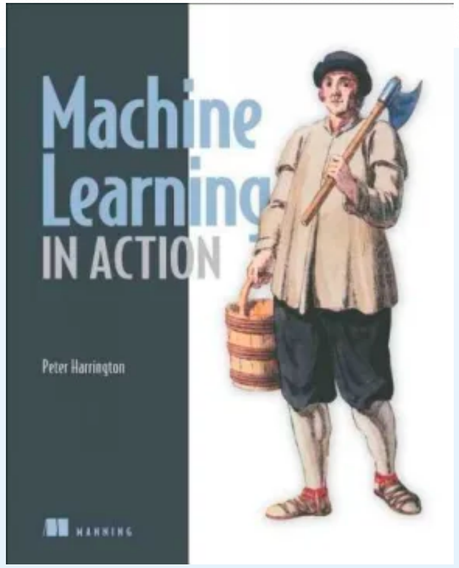
Recommended Textbooks

The textbooks are optional. Some of the content of this course is inspired from these books loosely

Optional Readings:

1. *Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow, 2nd Edition* (O'Reily). Copyright: September 2019. 10-day Trial Or free through UCLA Library
2. *Machine Learning in Action* (Peter Harrington). Copyright: April 3, 2012. ISBN : 9781638352457-1638352453





Github

What is GitHub?

An online repository that helps developers store, manage, track, share and control their code

Main benefits :

1.) Use other open source code
2.) Share Code with a Team
3.) Version Control

How necessary is GitHub?

The most popular source

It is almost a given that you know about GitHub in a Data role

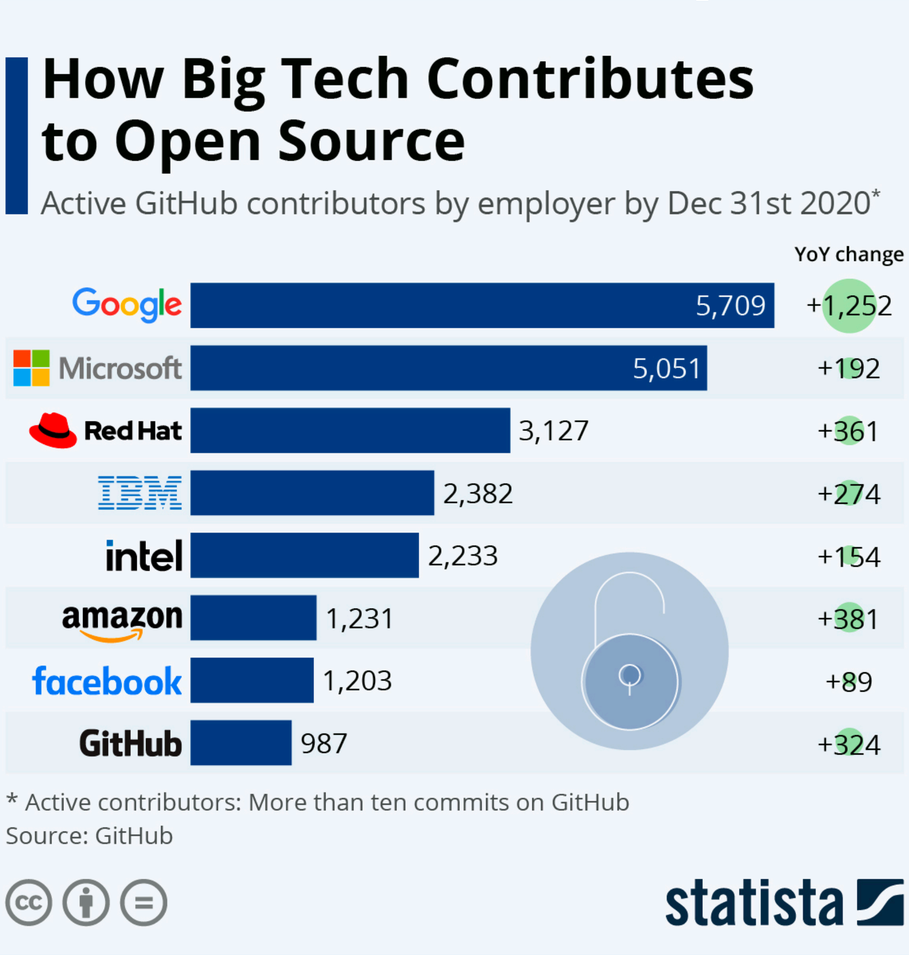
It is commonly used to host open source software development projects. As of June 2022, GitHub reported having over 83 million developers and more than 200 million repositories, including at least 28 million public repositories.

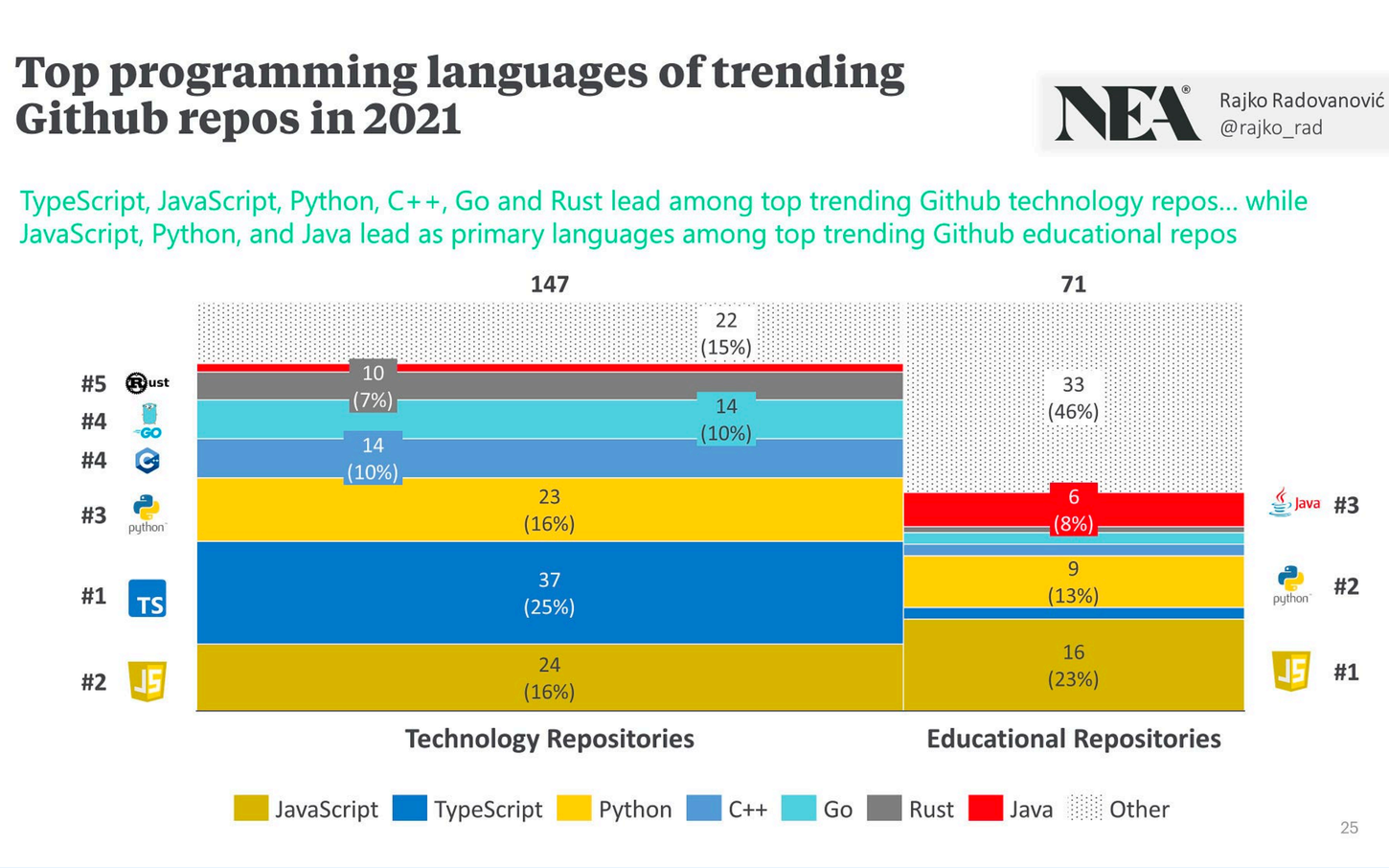
Of the 840 machine learning engineers in the study, **61%** said they use GitHub for sharing, the highest percentage of any profession in the report to do so. Jul 11, 2022

It is the largest source code host as of November 2021.

Source : https://thenewstack.io/where-do-data-practitioners-prefer-to-collaborate-github/#:~:text=Of%20the%20840%20machine%20learning,the%20report%20to%20do%20so.

How necessary is GitHub?

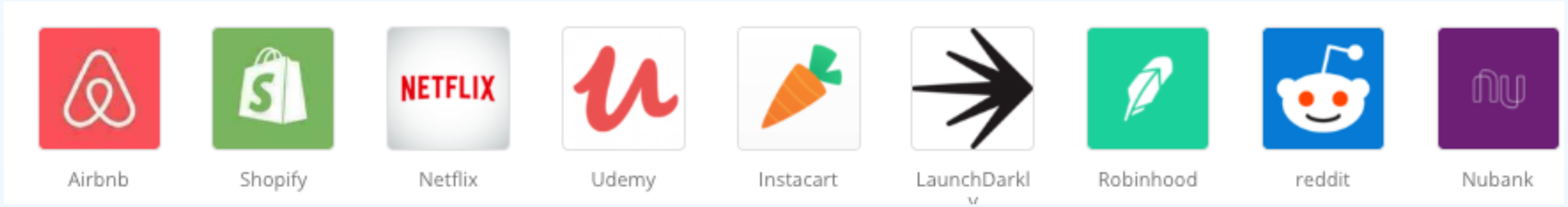




Companies that use GitHub?

Stack share reports

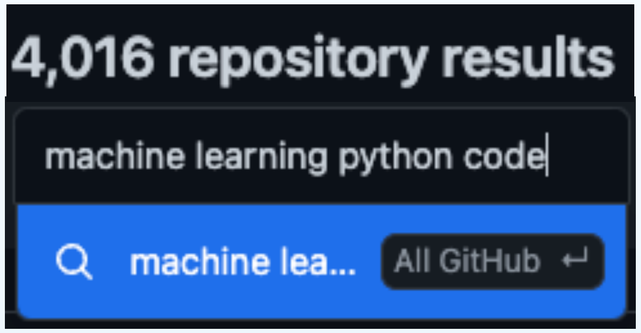
10029 companies reportedly use GitHub in their tech stacks, including Airbnb, Shopify, and Netflix.



1.) Use other open source code

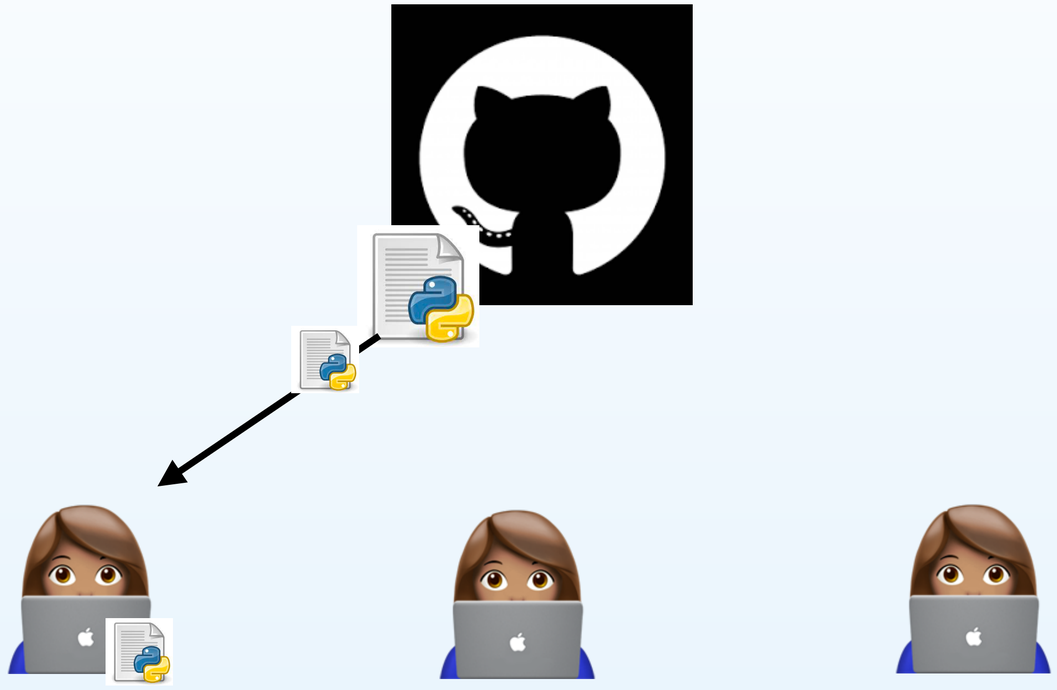
There is a huge amount of sample code you can learn from

This is an incredible resource



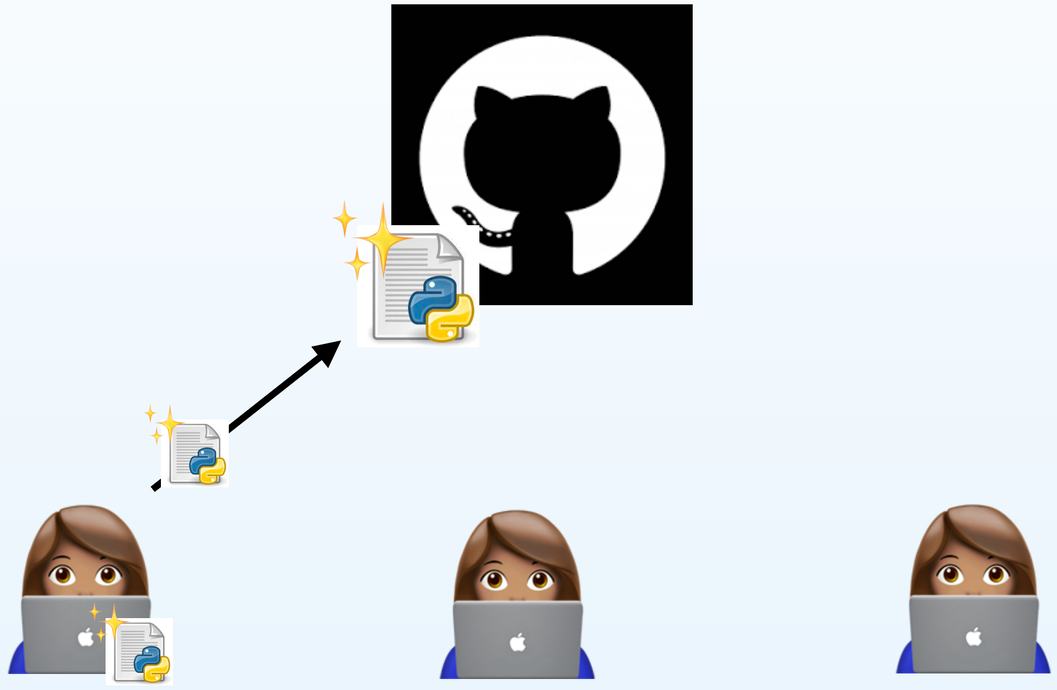
2.) Share code with a team

Github allows a team of data scientists to have a central location to pull the most recent code and post their most recent edits



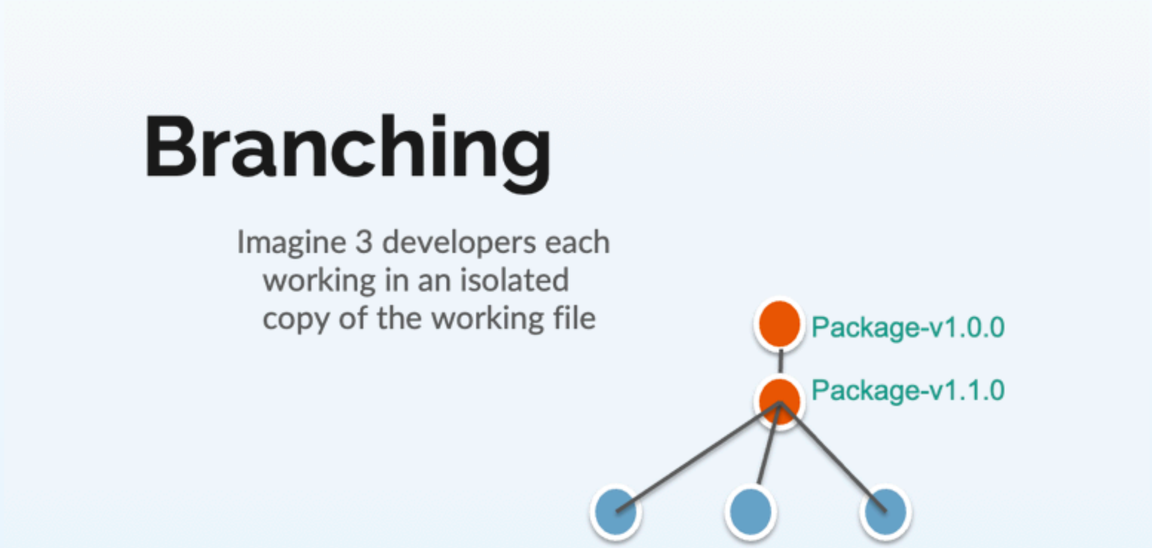
2.) Share code with a team

Github allows a team of data scientists to have a central location to pull the most recent code and post their most recent edits



3.) Version Control

Version Control : Tracking changes in any set of files, usually used for coordinating work among programmers collaboratively



3.) Version Control

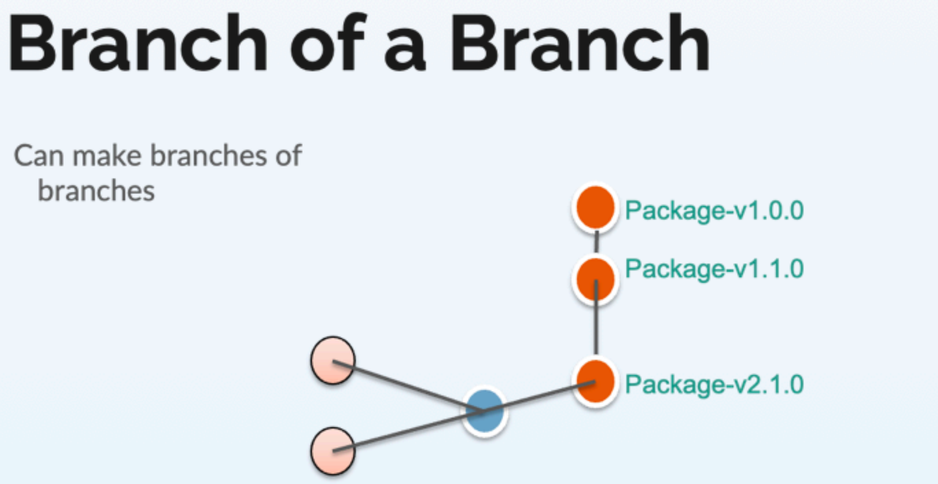
Version Control : Tracking changes in any set of files, usually used for coordinating work among programmers collaboratively

Merge

Changes are then merged into the develop branch after quality check

3.) Version Control

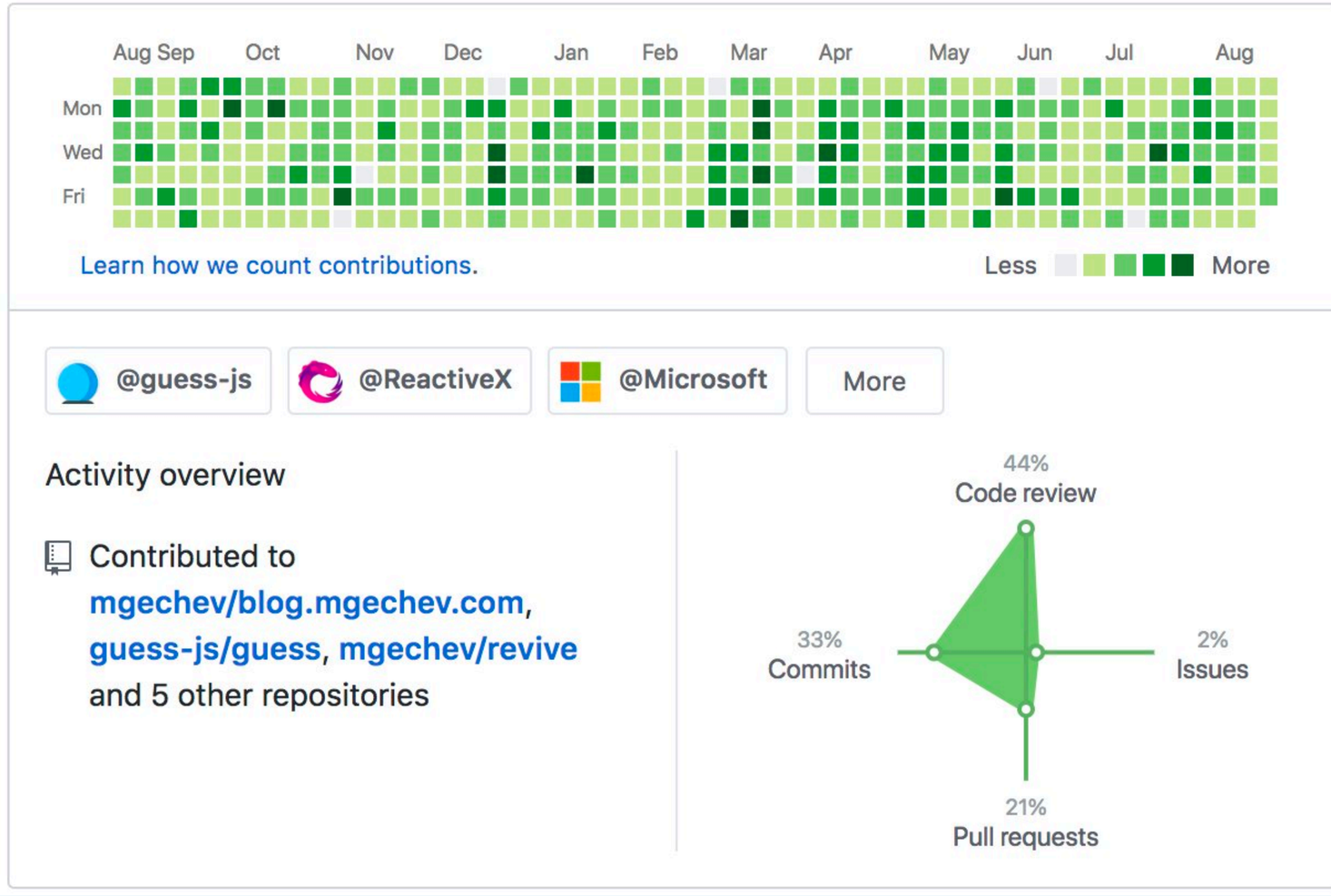
Version Control : Tracking changes in any set of files, usually used for coordinating work among programmers collaboratively



4.) Signal to Companies you are an Active developer

More and more often for data driven roles, employers will request your GitHub

Store your projects here so they can see your work

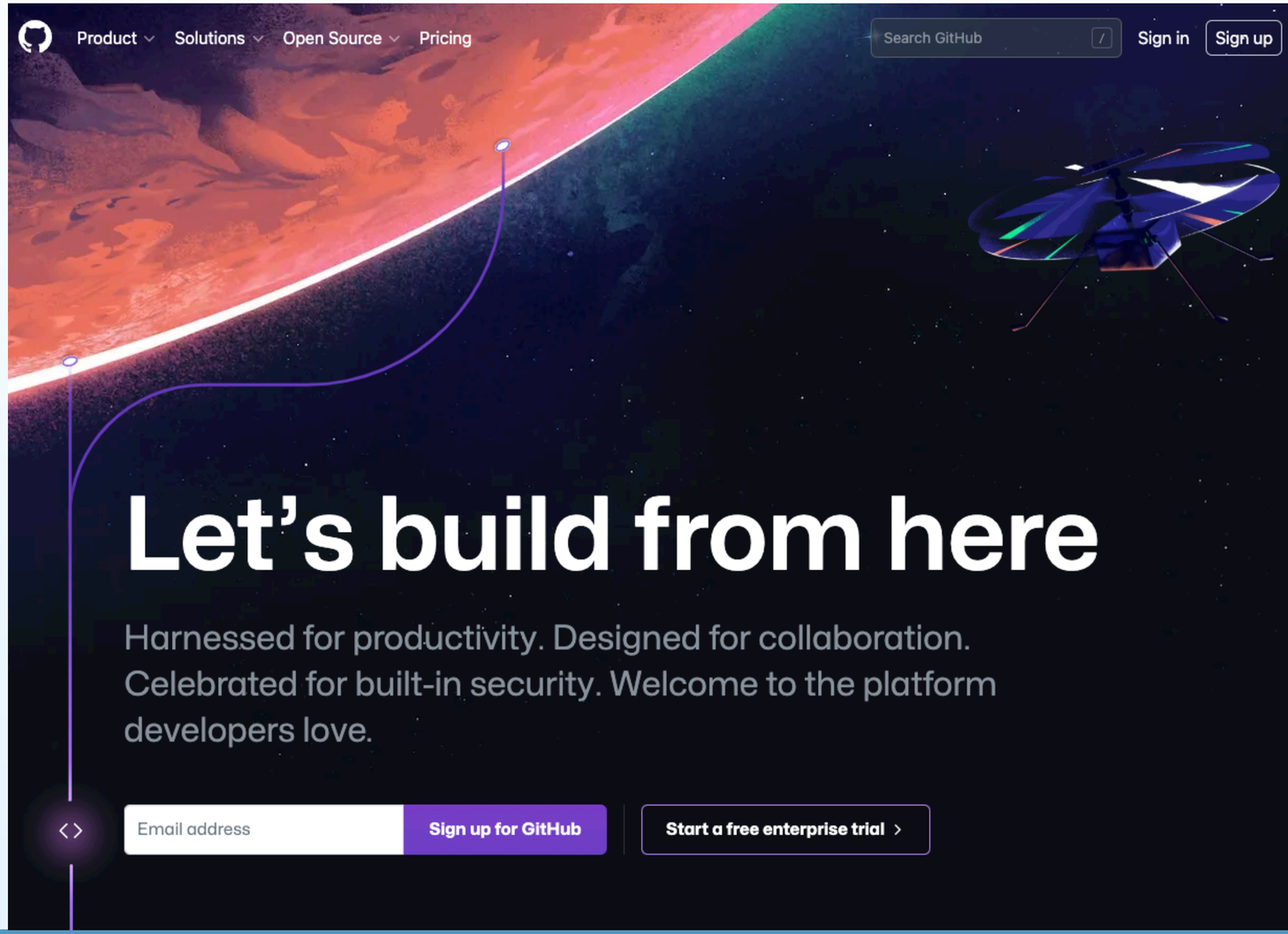


Make a GitHub Account

GitHub.com

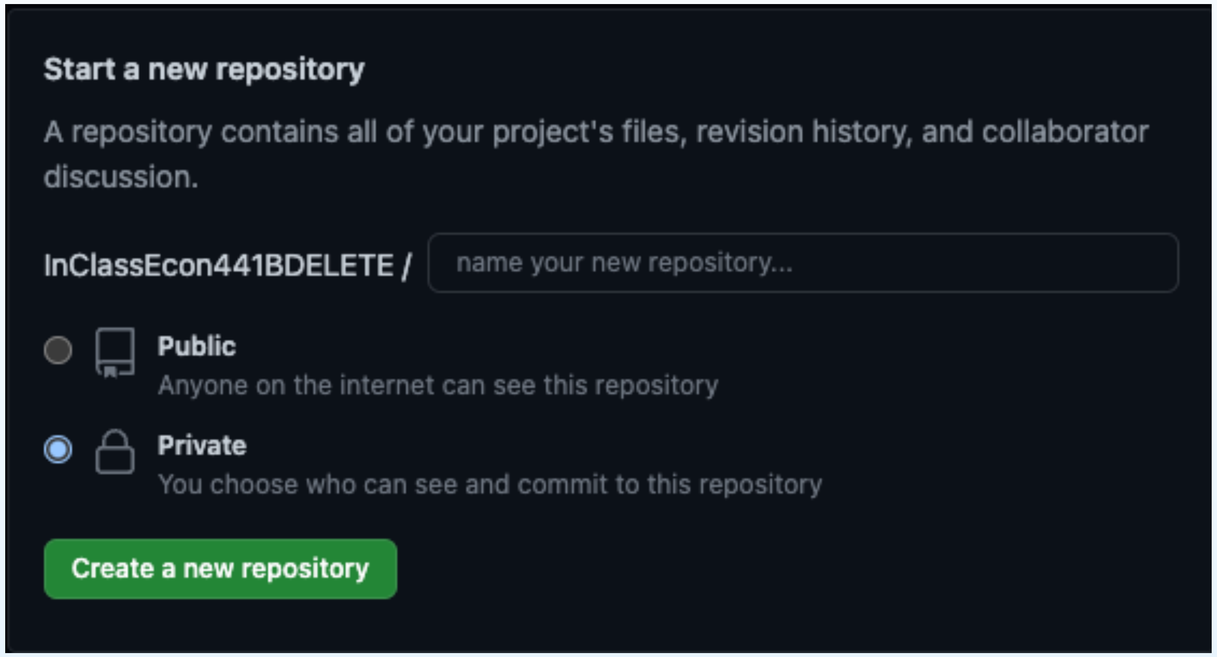
Username

InClassEcon441BDELETE



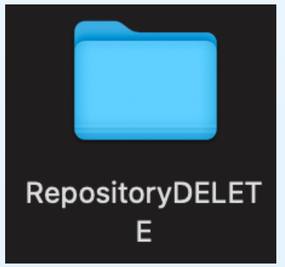
Make a Public Repository

GitHub.com



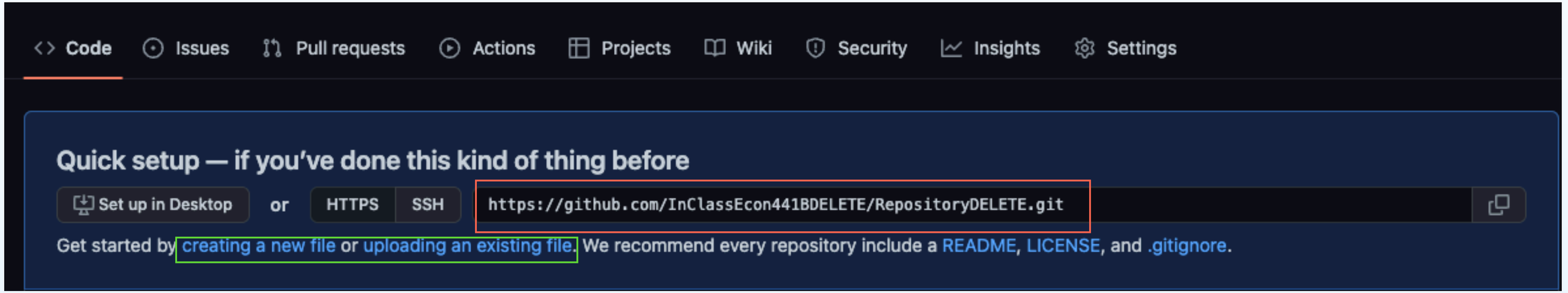
Name you repository

Repositories work the same as a folder on your computer



Adding to the repository

There are several ways to get code into a repository

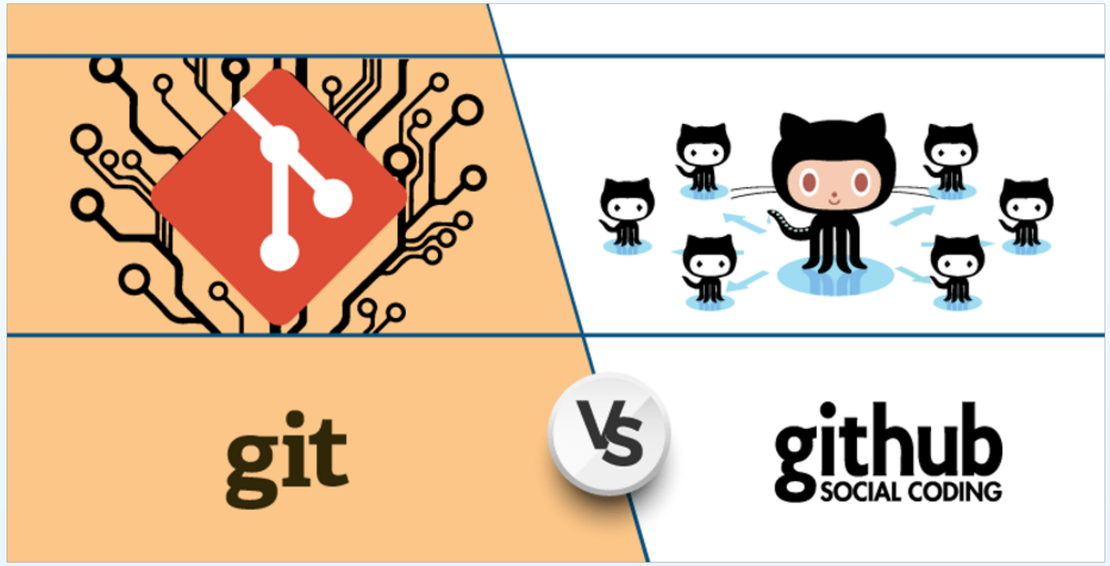


This is what you will submit to Bruin Learn at the end of class

This is how we will begin to upload files

What is Git?

Is the underlying code that Github lets visualize



Git is a version control system
Keeps track of your changes to code

A cloud based interface that
Helps visualize what Git is doing

What is Git?

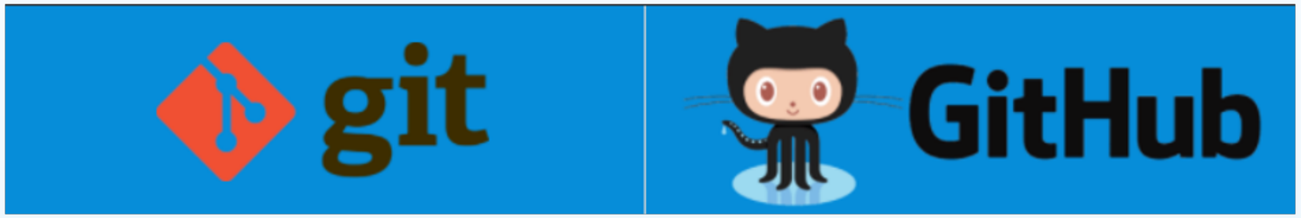
It is a software, on your local computer that allows you to do the version controlling and posting to Github
You can use GitHub with or without using Git

	
1. It is a software	1. It is a service
2. It is installed locally on the system	2. It is hosted on Web
3. It is a command line tool	3. It provides a graphical interface
4. It is a tool to manage different versions of edits, made to files in a git repository	4. It is a space to upload a copy of the **Git** repository
5. It provides functionalities like Version Control System Source Code Management	5. It provides functionalities of Git like VCS, Source Code Management as well as adding few of its own features

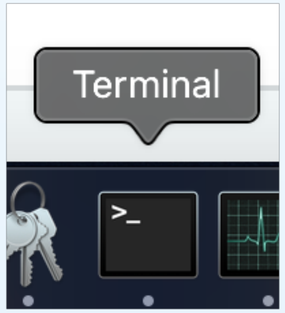
What is Git?

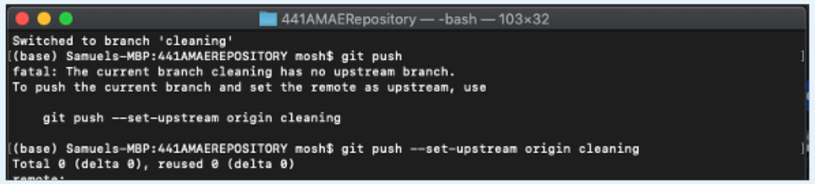
Git uses commands written in shell scripts

Accessible through



Terminal



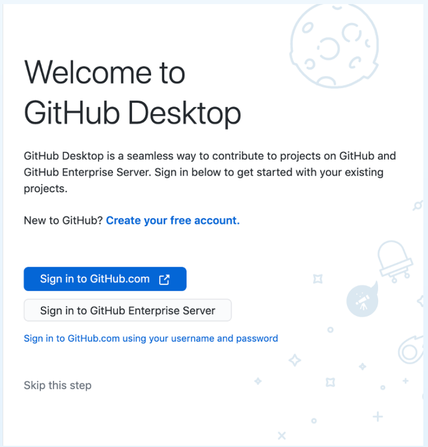


What is Git?

Github can be used without writing any code

There is a lot we do not cover about Git/Github

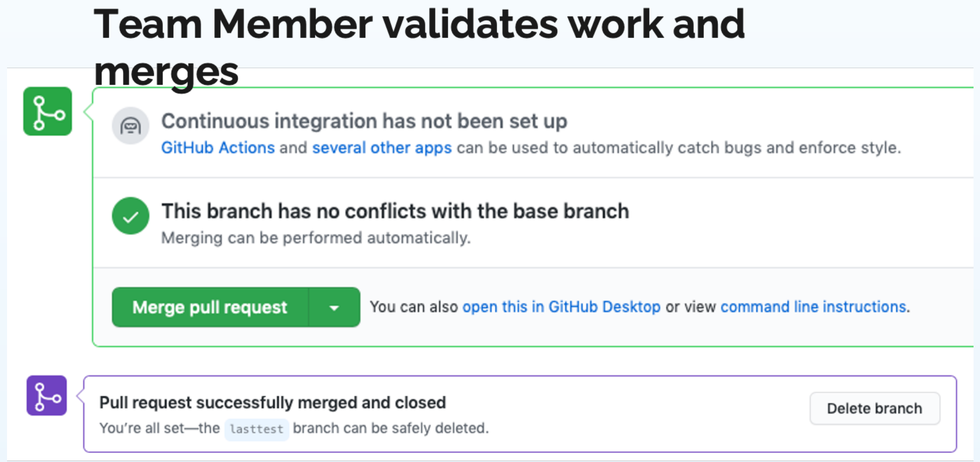
If this is something that interests you: Research and practice more

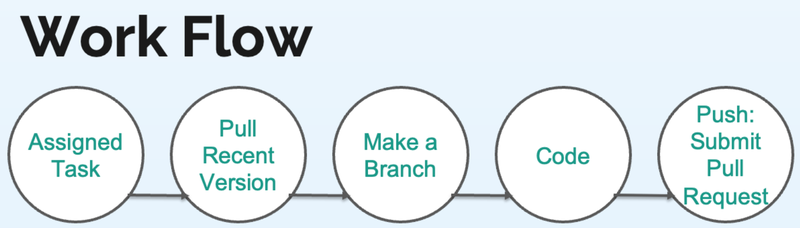


Go to:
mac.github.com

desktop.github.com

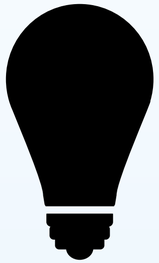
Step 1: Download Github desktop
Step 2: Make a Github account
Step 3: Make a Repository
3.a.) Posting on the Website
3.b.) Clone a repository
3.c.) Create a new repository
3.d.) Add existing Repository
Step 4: Get coding/editing





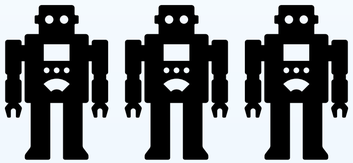
Robust Model Building

Machine Learning Pipeline

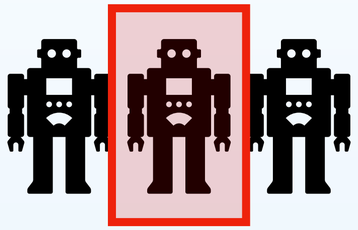


Have an Assumption : This data has predictive capabilities over the data

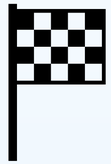
Data is randomly split into 3 parts : In-sample, Out of Sample and Hold out



Various models are built using In-Sample Data with data best practices such as Cross Validation



Final Model is selected based on Out-Of Sample performance



If the final model does not perform as expected on Hold-Out Datapractices such as Cross Validation. The assumption is thrown out

Machine Learning Pipeline : Example Regression

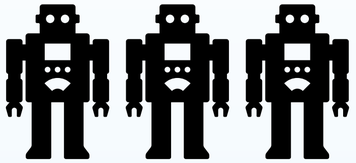
Have an Assumption :
This data has predictive capabilities over the data

Data is randomly split into 3 parts :
In-sample, Out of Sample and Hold out

MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22
7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24
5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25
3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25

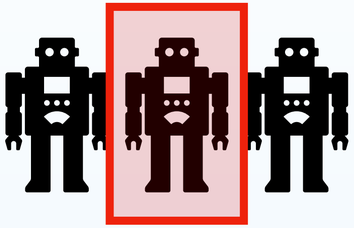
This Data Predicts Price

Machine Learning Pipeline : Example Regression

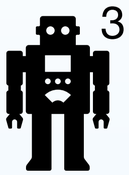


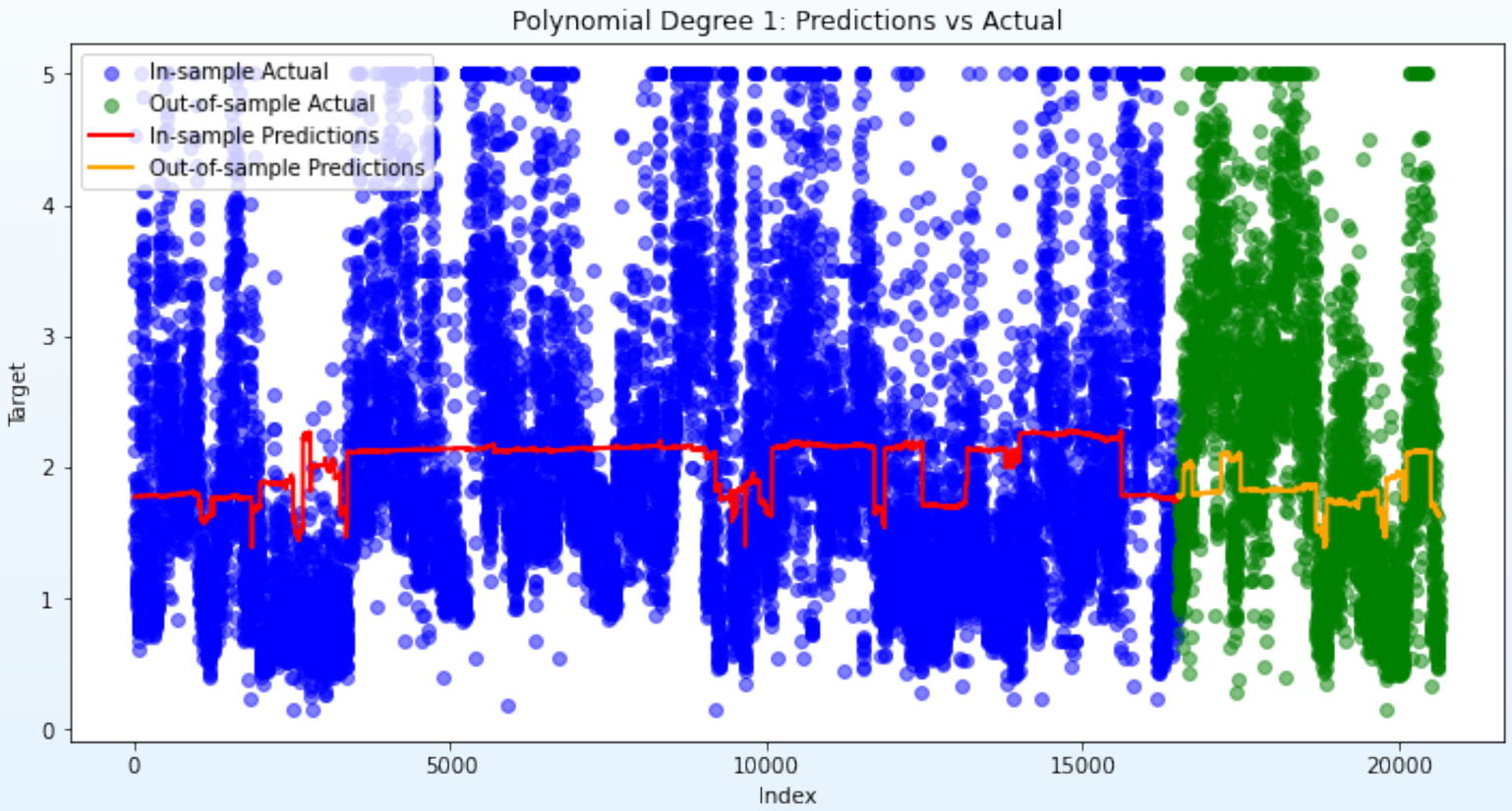
Various models are built using In-Sample Data with data best practices such as Cross Validation

Machine Learning Pipeline : Example Regression



Final Model is selected based on Out-Of Sample performance



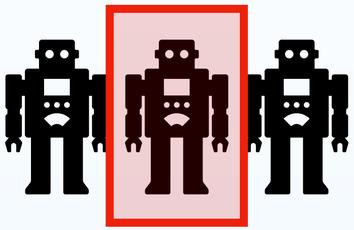


Bias (Underfitting)

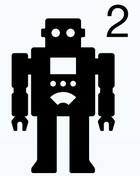
Bias refers to the error due to overly simplistic assumptions in the learning algorithm. A model with high bias tends to underfit the data, meaning it oversimplifies the underlying patterns. This leads to poor predictive performance as the model cannot capture the complexity of the real-world problem.

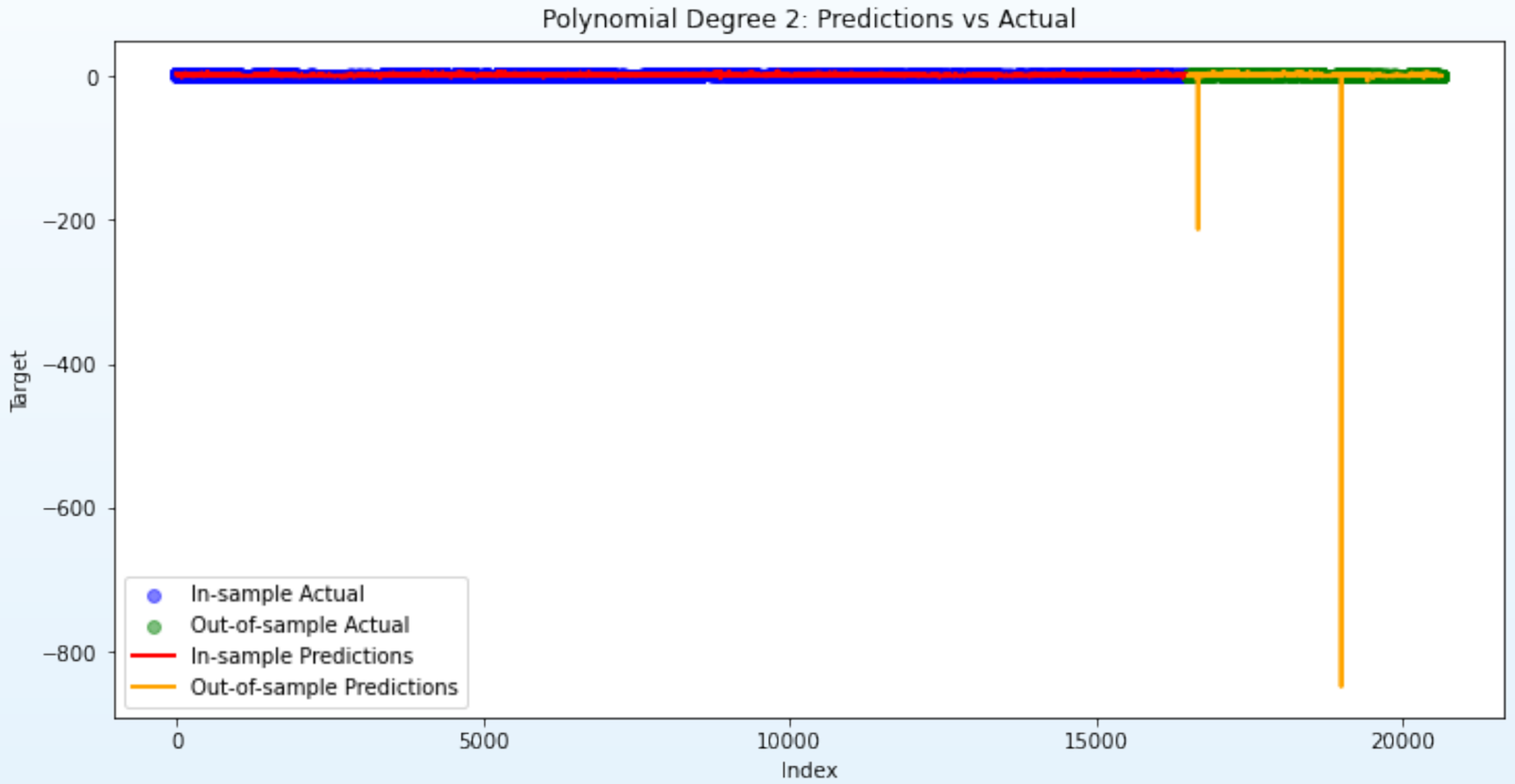
Example : Suppose you are training a linear regression model to predict housing prices. If the model assumes that the relationship between the features (e.g., square footage, number of bedrooms) and the price is strictly linear, it may perform poorly when the true relationship is more complex.

Machine Learning Pipeline : Example Regression



Final Model is selected based on Out-Of Sample performance



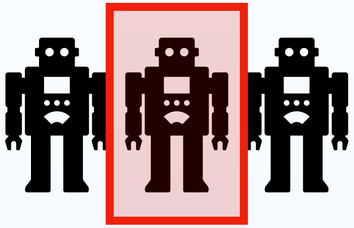


Variance (Overfitting)

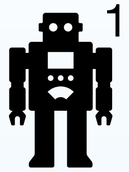
Variance, on the other hand, is the error due to excessive complexity in the learning algorithm. A model with high variance captures not only the underlying patterns but also the noise in the training data. This leads to poor generalization to unseen data.

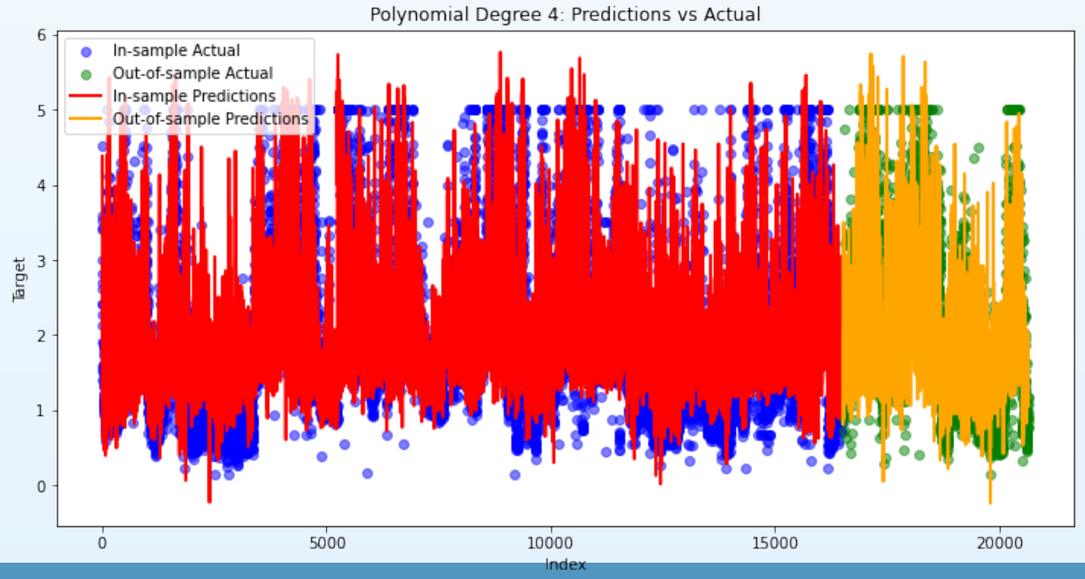
Example : Imagine training a decision tree with a very deep structure on a dataset of handwritten digits. While the tree can fit the training data perfectly, it might perform poorly on new, unseen digits because it has essentially memorized the training examples, including their individual quirks.

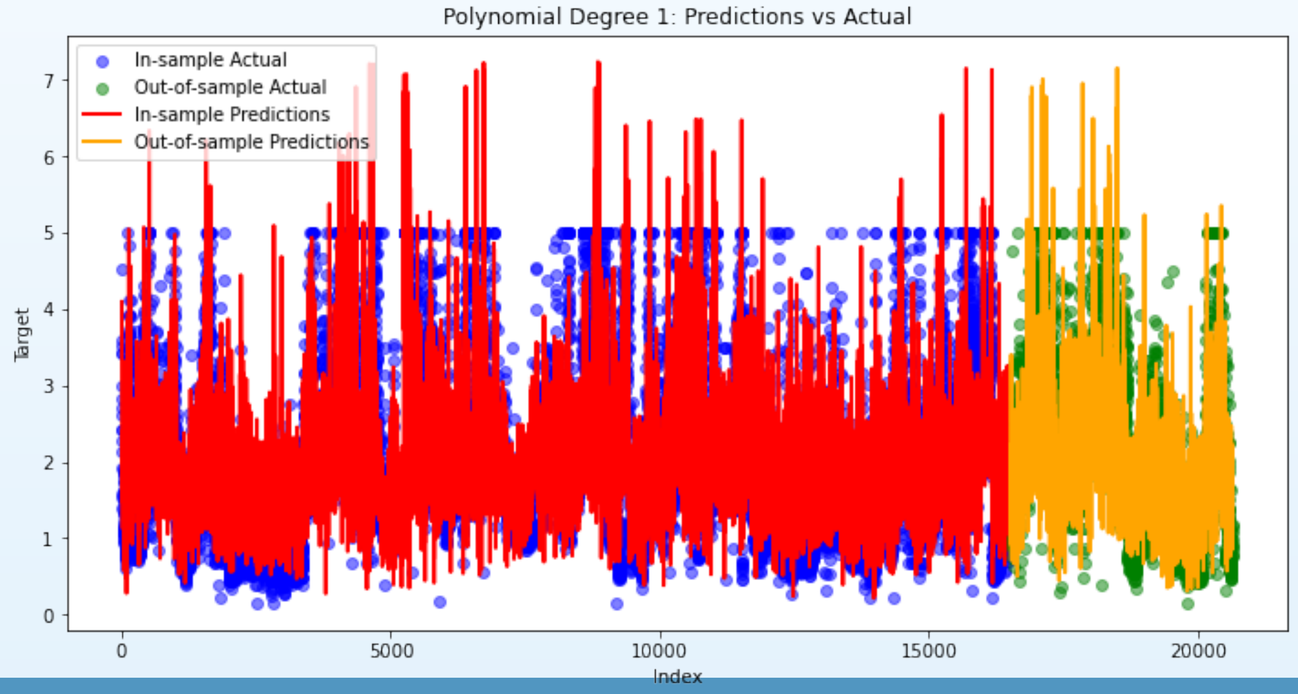
Machine Learning Pipeline : Example Regression



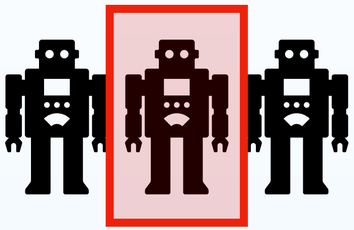
Final Model is selected based on Out-Of Sample performance





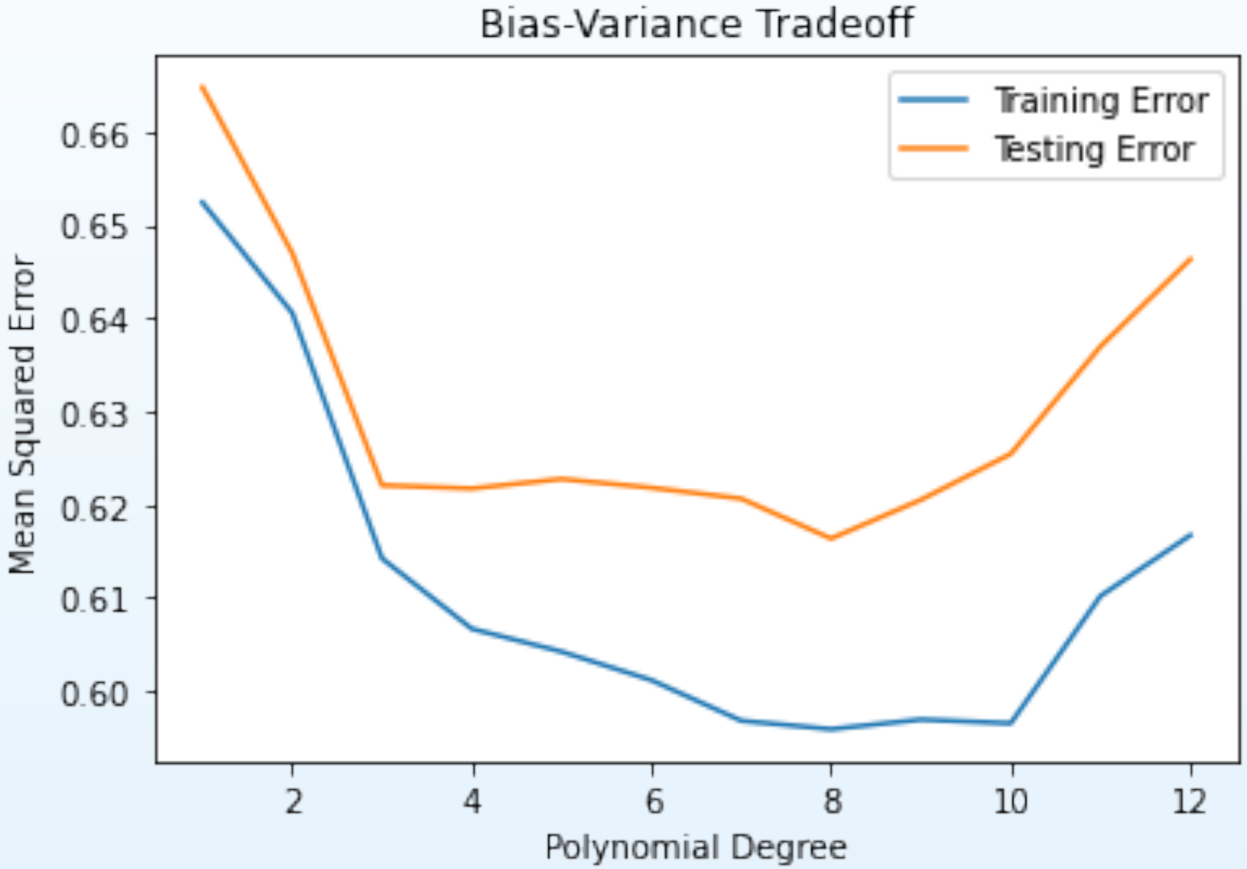


Machine Learning Pipeline : Example Regression



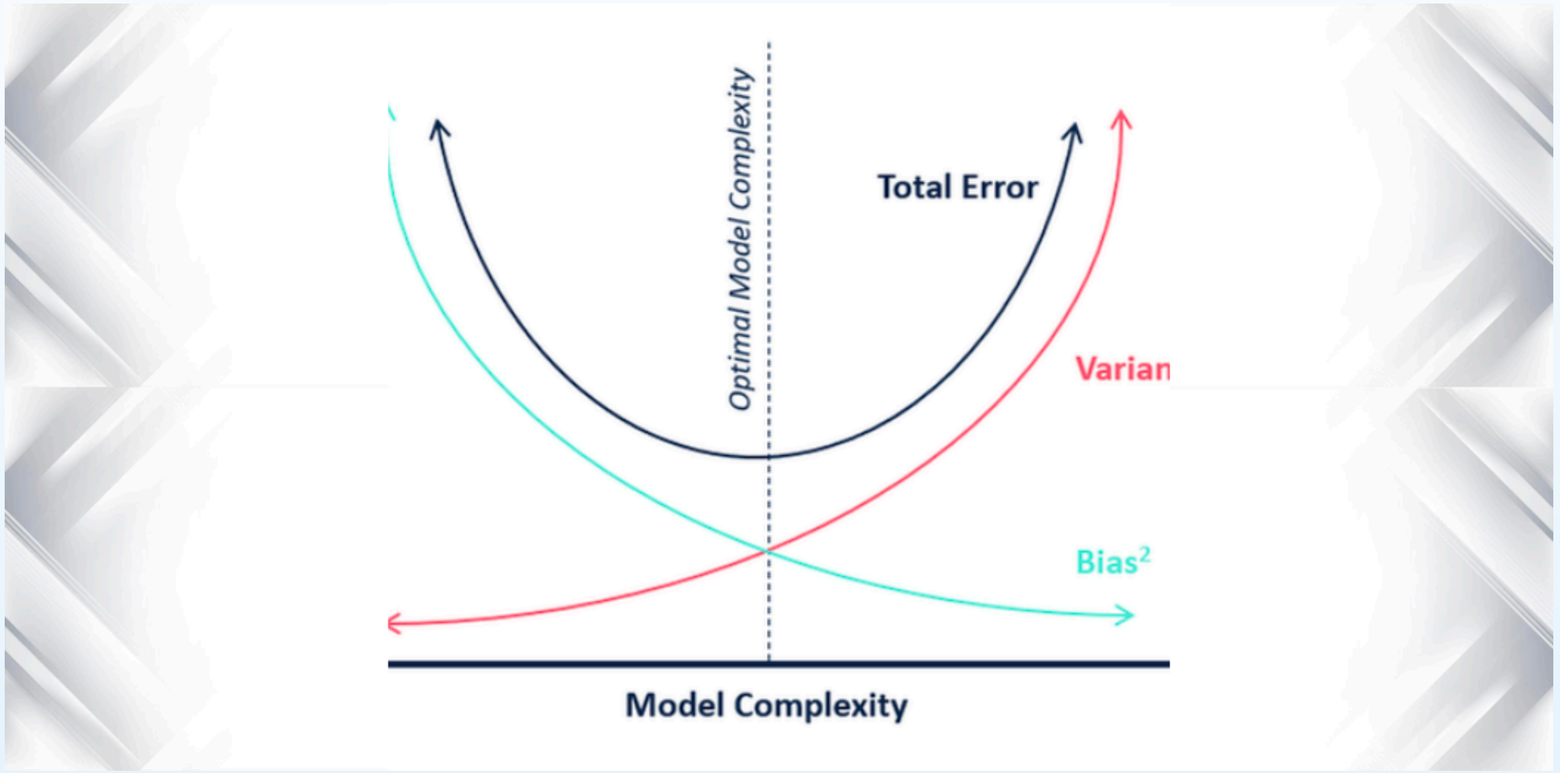
Final Model is selected based on Out-Of Sample performance

What is happening here?
As model complexity goes up we start getting higher Variance



Measuring Bias Variance Trade-off

Always have to deal with the Bias-Variance trade off. We will learn models to optimize automatically

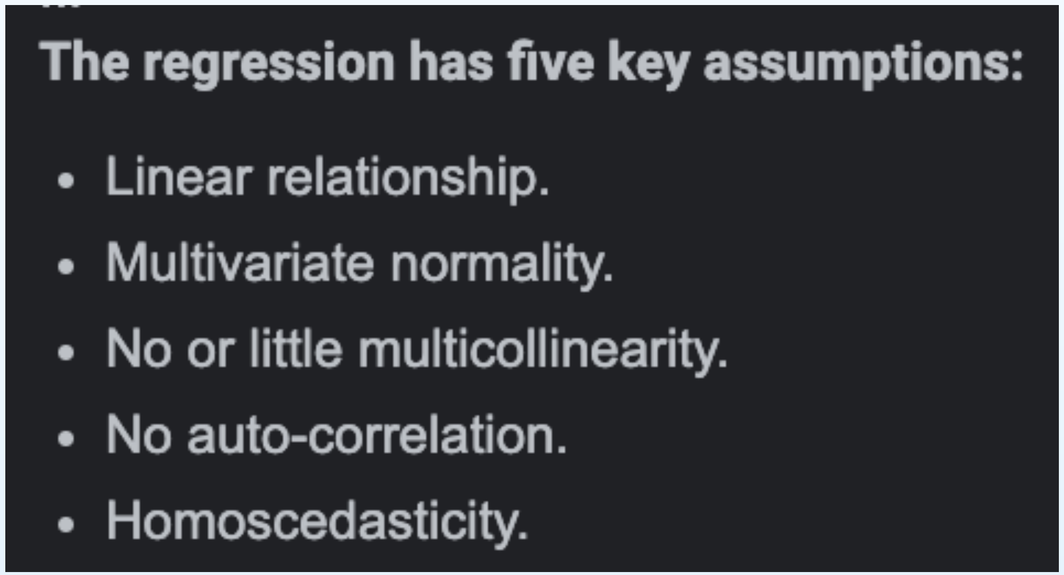


Linear Regression with StatsModels

Linear Regression

Have to meet the Gauss-Markov assumptions

We will not go through these in this class because you have done so much in 430



A python package that has prebuilt statistical test

- Regression and Linear Models
  - Linear Regression
    - Module Reference
    - Generalized Linear Models
    - Generalized Estimating Equations
    - Generalized Additive Models (GAM)
    - Robust Linear Models
    - Linear Mixed Effects Models
    - Regression with Discrete Dependent Variable
    - Generalized Linear Mixed Effects Models
    - ANOVA
    - Other Models othermod
  - Time Series Analysis
  - Other Models
  - Statistics and Tools
  - Data Sets

statsmodels

We use scikit learn mainly in this class. So why stats models ever?

Mainly, their regression outputs

```
In [7]: print(res.summary())
                            OLS Regression Results
==============================================================================
Dep. Variable:                  GRADE   R-squared:                       0.416
Model:                            OLS   Adj. R-squared:                  0.353
Method:                 Least Squares   F-statistic:                     6.646
Date:                Thu, 04 Jan 2024   Prob (F-statistic):            0.00157
Time:                        18:30:41   Log-Likelihood:                -12.978
No. Observations:                  32   AIC:                             33.96
Df Residuals:                      28   BIC:                             39.82
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
GPA            0.4639      0.162      2.864      0.008       0.132       0.796
TUCE           0.0105      0.019      0.539      0.594      -0.029       0.050
PSI            0.3786      0.139      2.720      0.011       0.093       0.664
const         -1.4980      0.524     -2.859      0.008      -2.571      -0.425
==============================================================================
Omnibus:                        0.176   Durbin-Watson:                   2.346
Prob(Omnibus):                  0.916   Jarque-Bera (JB):                0.167
Skew:                           0.141   Prob(JB):                        0.920
Kurtosis:                       2.786   Cond. No.                         176.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
```

Python Refresher

Object

Anything that can be
Stored as a variable
Name

```
df
```

	0
0	24.0
1	21.6
2	34.7

Parameters

Creating an object or function,
a user input
<func>(, <parameter> = <value>)

```
pd.DataFrame(target,
 columns = ["Target"])
```

Methods

A function based
For a specific object type
<object>.<method>( )

```
df.shift()
```

	0
0	NaN
1	24.0
2	21.6

Attributes

An object associated
with another object
<object>.<attribute>

```
df.columns
```

```
RangeIndex(start=0,
```

Splitting the Data

Pandas DataFrame

```
1 d
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	0
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422
...	...	...	...	...	...	...	...	...	...
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	-121.09	0.781
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	-121.21	0.771
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	-121.22	0.923
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	-121.32	0.847
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	-121.24	0.894

20640 rows × 9 columns

Randomly Sampled

```
1 d.sample(len(d))
```

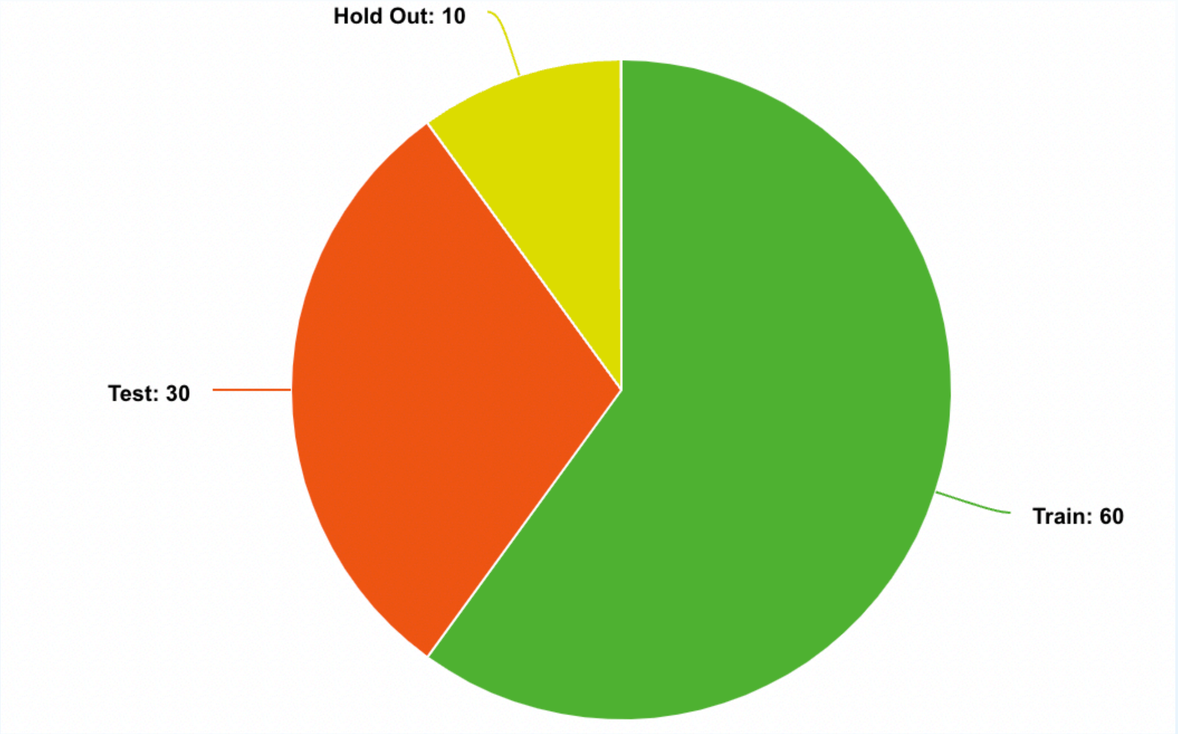
	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	0
6364	6.4576	33.0	5.661327	1.080092	1268.0	2.901602	34.15	-117.97	5.00001
1181	2.2045	41.0	5.122222	1.188889	284.0	3.155556	39.48	-121.55	0.41800
13742	10.8634	20.0	5.903448	0.855172	395.0	2.724138	34.03	-117.19	3.81800
629	3.3710	42.0	4.530120	0.992470	1817.0	2.736446	37.72	-122.17	1.65000
3858	11.2093	42.0	6.876106	0.902655	284.0	2.513274	34.17	-118.43	5.00001
...	...	...	...	...	...	...	...	...	...
10345	5.7871	24.0	6.184295	1.059295	1972.0	3.160256	33.61	-117.67	2.27400
5001	1.7849	46.0	4.294326	1.138298	1153.0	4.088652	33.99	-118.28	0.99300
15172	5.7382	17.0	6.444043	0.920578	667.0	2.407942	33.04	-117.06	2.78000
11318	3.4955	15.0	3.709447	0.946524	1617.0	2.882353	33.77	-117.99	1.60900
19869	3.1486	14.0	5.218341	0.960699	568.0	2.480349	36.33	-119.35	1.05600

20640 rows × 9 columns

Splitting the Data

```
split1 = int(len(df) * .6)
split2 = int(len(df) * .9)
d_in = d[:split1]
d_out = d[split1:split2]
d_hold = d[split2:]
```

```
X_in = d_in.iloc[:,:-1]
y_in = d_in.iloc[:,-1:]
```



Running the Model

Build your first Model

INPUT TESTING OF GUASS-MARKOV ASSUMPTIONS

```
X_in = sm.add_constant(X_in)
model1 = sm.OLS(y_in, X_in).fit()
```

```
dir(model1)
```

```
'aic',
'bic',
'bse',
'centered_tss',
'compare_f_test',
'compare_lm_test',
'compare_lr_test',
'condition_number',
'conf_int',
'conf_int_el',
'cov_HC0',
'cov_HC1',
'cov_HC2',
'cov_HC3',
'cov_kwds',
'cov_params',
'cov_type',
'df_model',
'df_resid',
'eigenvals',
'el_test',
'ess',
'f_pvalue',
'f_test',
'fittedvalues',
'fvalue',
'get_influence',
'get_prediction',
'get_robustcov_results',
'info_criteria',
'initialize',
'k_constant',
'llf',
'load',
'model',
'mse_model',
'mse_resid',
'mse_total',
'nobs',
'normalized_cov_params',
'outlier_test',
'params',
'predict',
'pvalues',
'remove_data',
'resid',
'resid_pearson',
'rsquared',
'rsquared_adj',
'save',
'scale',
'ssr',
'summary',
'summary2',
't_test',
't_test_pairwise',
'tvalues',
'uncentered_tss',
'use_t',
'wald_test',
'wald_test_terms',
'wresid']
```

Analyzing the Model

Build your first Model

```
1 print(model1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      0   R-squared:                       0.589
Model:                            OLS   Adj. R-squared:                  0.589
Method:                 Least Squares   F-statistic:                     2216.
Date:                Wed, 10 Jan 2024   Prob (F-statistic):               0.00
Time:                        11:31:26   Log-Likelihood:                -13748.
No. Observations:               12384   AIC:                         2.751e+04
Df Residuals:                   12375   BIC:                         2.758e+04
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        -28.7672      1.004    -28.641      0.000     -30.736     -26.798
MedInc         0.4330      0.005     80.271      0.000       0.422       0.444
HouseAge       0.0056      0.001      9.020      0.000       0.004       0.007
AveRooms      -0.0949      0.008    -12.531      0.000      -0.110      -0.080
AveBedrms      0.5781      0.036     16.135      0.000       0.508       0.648
Population -1.546e-05   6.34e-06     -2.437      0.015   -2.79e-05   -3.02e-06
AveOccup      -0.0080      0.001     -6.659      0.000      -0.010      -0.006
Latitude      -0.4003      0.011    -36.586      0.000      -0.422      -0.379
Longitude     -0.3611      0.011    -31.449      0.000      -0.384      -0.339
==============================================================================
Omnibus:                     2751.880   Durbin-Watson:                   0.842
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             8334.861
Skew:                           1.147   Prob(JB):                         0.00
Kurtosis:                       6.301   Cond. No.                     2.76e+05
==============================================================================
```

```
1 model1.rsquared_adj
```

```
0.588697421105419 9
```

```
1 model1.mse_total
```

```
1.3119119613283674
```

```
1 model1.params
```

```
const       -28.767223
MedInc        0.432963
HouseAge      0.005554
AveRooms     -0.094948
AveBedrms     0.578070
Population   -0.000015
AveOccup     -0.007995
Latitude     -0.400267
Longitude    -0.361127
```

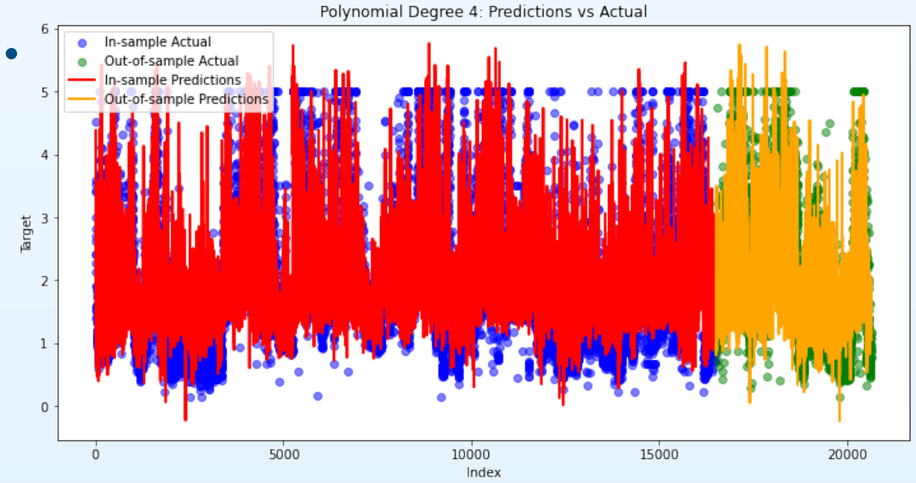
Test in Out of Sample

Pick your measure. Which ever does the best in out of sample is your selected model

```
out_pred = model1.predict(X_out)
mean_squared_error(y_out, out_pred)
```

In-Class Assignment

0.) Make a Jupyter notebook
1.) Import from FRED Data
2.) Do not randomize, split the data into Train test hold out
3.) Build a model that regresses FF~Unemp, HousingStarts, Inflation
4.) Recreate this graph for your model
5.) What is the in and out of Sample MSE.



6.) Using a for loop. Repeat 3,4,5 for polynomial degrees 1,2,3
7.) State your observations