**LunarTech Machine learning**

**Supervised vs. Unsupervised**

* **Supervised:** Requires training data with independent variables & a dependent variable (labelled data)
  + Need labelled data that can “supervise” the algorithm when learning from the data
    - Regression Models
    - Classification Models
* **Unsupervised**
  + Requires training data with independent variables only
  + No need labelled data that can “supervise” the algorithm when learning from the data
    - Clustering Models
    - Outlier Detection Models

**Regression Vs. Classification**

* **Regression**
  + Can be used when the response variable to be predicted is a continuous variable (scaler)
  + Examples: Linear Regression, Fixed Effects Regression, XGBoost Regression
* **Classification**
  + Can be used when the response variable to be predicted is a continuous variable (scaler)
  + Examples: Logistic Regression, XGBoost Classification

**Regression Performance Metrics**

* **RSS:** Residual sum squared ( SUM (actual\_values – predicted\_values)^2 )
* **MSE:** Mean squared error ( AVG [ SUM (actual\_values - predicted\_values)^2 ] )
* **RMSE:** Root mean squared error ( SQRT {( AVG [ SUM (actual\_values - predicted\_values)^2 ] } )
* **MAE:** Mean squared error ( AVG [ SUM ( ABS (actual\_values – predicted\_values) ) ] )
* **A screenshot of a math equation

  Description automatically generated**
* MAE: Lower value represents a better fit
* MSE: Commonly used if you want to penalize large errors more than the small ones
  + Sensitive to outliers or extreme values
* RMSE: Easier to interpret because it’s in the same units as Target variable
  + Commonly used when you want to compare the performance of different models or when you want to report the error in a way that it is easier to understand
* MAE: Commonly used to penalize all errors equally regardless of their magnitude
  + Less sensitive to outliers compared to MSE

**Classification Performance Metrics**

* **Accuracy:** 
  + A proportion of correct predictions made by the model
  + Calculated by taking the correct predictions and divided by all number of predictions
* **Precision**:
  + A proportion of true positive predictions divided by all positive predictions
  + True positives are cases where the model correctly predicts a positive outcome
  + False positives are cases where the model incorrectly predicts a positive outcome
* **Recall**:
  + A proportion of true positive prediction divided by all actual positive instances
* **F-1 Score**: =
  + Harmonic name or usual mean of the precision and recall
  + Higher value indicates a better balance between precision and recall
* Performance metrics measure the ability of the machine learning model to correctly classify instances into the correct categories

**Clustering Performance Metrics**

* Performance evaluated using metrics that measure the similarity of the data points within a cluster and the dissimilarity of the data points between different clusters
* **Homogeneity:** 
  + The measure of the degree to which all of the data points within a single cluster belong to the same class
  + A higher value indicates a more homogeneous cluster
* **Silhouette Score:** 
  + The measure of the similarity of the data point to its own cluster compared to the other clusters
  + A higher silhouette score indicates that the data point is well matched to its own cluster
  + Usually used for DB Scan or K means
  + S(o): Silhouette coefficient of the data point
  + A(o): Average distance between {o} and all the other data points in the cluster to which {o} belongs to
  + B(o): Minimum average distance from {o} to all the Clusters to which {o} does not belong to
* **Completeness:** 
  + The measure of the degree to which all of the data points that belong to a particular class are assigned to the same cluster
  + A higher value indicated a more complete cluster

**Training Machine Learning Model**

* **Data Preparation:**
  + Split the data into train, validation and test.
  + **Training set:** Used to feed the model
  + **Validation set:** Used to optimize your hyperparameters and to pick the best model
  + **Test set:** Used to evaluate the model’s performance
* **Model Training:**
  + Train the model on the training data and save the fitted model
  + Can you a algorithm or set of algorithms
* **Hyper-Parameter Tuning:**
  + Use the fitted Model and Validation Set to find the optimal set of parameters where the model performs the best
  + Need to adjust the model’s parameters to minimize the error on the training set by performing hyperparameter tuning
    - To do hyperparameter tunning: Need to use validation data and then we can select the best model that results in the least possible validation error rate
* **Prediction:**
  + Use the optimal set of parameters from Hyper-Parameter Tuning Stage and training data, to train the model again with these hyper parameters, use this best fitted model to predictions on test data.
* **Test Error Rate**
  + Compute the performance metrics for your model using the predictions and real values of the target variable from your test data
* **Training Dataset:** This is the dataset used to train the model. It’s where the model learns the patterns in the data. The goal is to adjust the model’s parameters so that it can make accurate predictions on this data. The training dataset is used to fit the model to the data.
* **Validation Dataset:** The validation dataset is used during the model development process, specifically after the model has been trained on the training dataset. It’s used to tune the model’s hyperparameters, such as the learning rate, number of layers in a neural network, and the number of trees in a random forest. The validation dataset provides a way to evaluate the model’s performance and adjust its configuration without using the test dataset. This helps to prevent overfitting, where the model performs well on the training data but poorly on new, unseen data.
* **Test Dataset:** The test dataset is used to evaluate the final model after all hyperparameters have been tuned and the model architecture is finalized. It’s the first time the model sees this data during the entire process. The test dataset provides an unbiased evaluation of how well the model can generalize to new, unseen data. The performance on the test dataset is what we ultimately care about when assessing the model’ effectiveness.
* **Summary:** The validation dataset is crucial because it provides a check on the model’s performance during the training process. It helps to ensure that the model is not overfitting the training data by providing a separate set of data to evaluate the model’s ability to generalize. The test dataset, on the other hand, is used to evaluate the model’s final performance and is not used during the training or validation process.

**Bias-Variance Trade Off**

* **Bias:** Bias of Machine Learning model is its inability to capture the true relationship in the data, mathematically equal to the difference between the Expectation of model estimation and its true value
  + Mahcine learning models that are able to detect the true relationship in the data have low bias
  + Usually complex models or more flexible models tend to have a lower bias than simpler models.
  + Mathematically the bias of the model can be expressed as the expectation of the difference between the estimate and the True Value
  + **A black text on a white background

    Description automatically generated**
* **Variance:** Variance of Machine Learning model is the inconstancy level of model performance when applying it to different data sets.
  + When the same model that is trained using training data performs entirely different than on test data then model variance is high
  + Complex models or more flexible models tend to have a higher variance than simpler models
* **Error:** Error of Machine Learning model assuming model is trained on (X1, Y1) (X2, Y2) … (Xn, Yn) to estimate the value of Y0.
* **Prediction Error Rate:**
  + A math equations on a white background

    Description automatically generated
* **Bias-Variance Trade off:** In order to minimize the expected test error rate, we need to select a Machine Learning method that simultaneously achieves low variance and low bias
  + Negative correlation between Variance and Bias of model
  + ML model’s flexibility has direct impact on its Variance / Bias
  + As flexibility of the model increases the model finds the true patterns in the data easier which reduces the bias of the model, but increases the variance
  + As the flexibility of the model decreases the model finds it more difficult to find the true patterns in the data which then increase the bias of the model, but decreases the variance
* A green sign with white rectangles and arrows

  Description automatically generated

**Overfitting vs. Regularization**

* **Overfitting:** Occurs when the model performs well in the training (low train error rate) while performs worse on the test data(high test error rate)
* **A diagram of a model error rate

  Description automatically generated**
* Ideal would be test error rate to be low or at least that the training error rate is equal to the test error rate
* Overfitting is where the model learns the detail and noise in training data to the point where it negatively impacts the performance of the model on this new data
  + As a consequence, the model follows the data too closely, closer than it should this means that the noise or random fluctuation of the training data is picked up and learned as concepts by the model which is should actually ignore
  + The problem is that the noise or random component of the training data will be very different from the noise in the new data. The model will therefore be less effective in making predictions on new data.
  + Overfitting is causes by having too many features | too complex of a model | or too little of data
  + A model with high variance and low bias usually results in a model where the flexibility is high, resulting in overfitting, due to the higher risk of having a model follow the data too closely and following the noise
* Underfitting is where our test error rate is much lower than our training error rate

**How to fix Overfitting?**

* **Reduce the complexity of the model**
  + Higher Complexity 🡪 Higher likelihood of following the data too closely, including noise
  + Reducing the flexibility of the model will also reduce overfitting as well by using a simpler model with fewer parameters or by applying a regularization technique such as L1 or L2 regularization
* **Collect more data**
  + The more data you have the less likely your model will overfit
* **Use Resampling Technique (CV)**
  + Cross-Validation: Technique that allows you to train and test your model on different subset of your data, which can help you to identify if your model is overfitting
* **Early Stopping**
  + Technique where you monitor the performance of the model on a validation set during the training process and stop the training when the performance starts to decrease
* **Ensemble Methods** 
  + Combining multiple models such as decision trees
* **Dropout**
  + A regularization technique for reducing overfitting in neural networks by dropping out or setting to zero some of the neurons during the training process

**Regularization**

* **Regularization:** Or Shrinkage is a method that shrinks some of the estimated coefficients towards zero, to penalize unimportant variables for increasing the variance of the model
  + Used solve the overfitting problem
  + Introduces a little bias in the model to decrease its variance 1:14

\*\*\*\*\*\*\* BREAK \*\*\*\*\*\*\*

**Business will come to you with Data and ask that we want to understand what features within this data have the biggest influence on this specific factor.**

* If we are dealing with aggregated data, so averages or block/groups of data
  + Use median if data is skewered
* **Linear Regression:** 
  + **Predictive:**  Can use linear regression for predictive analytics (Sklearn)
  + **Causation:** Can use linear regression for casual analysis so to identify and interpret the independent variables that have a statistically significant impact on your response variable (statsmodel.api as sm)

**Case Study:** Californian house prices

* **Independent variables:** 
  + Longitude
  + Latitude
  + Housing\_median\_age
  + Total\_rooms
  + Total\_bedrooms
  + Population
  + Households
  + Median\_income
  + Ocean\_proximity
* **Dependent variables:**
  + Median\_house\_value
* **Note:** 
  + What linear regression tries to use during causal analysis is that it tries to keep all the independent variables constant and then investigates for a specific independent variable what is this one unit change increase in the specific independent variable will result in what kind of change in our dependent variable
  + For instance if we change by one unit the housing median age 🡪 then what will the corresponding change in our median household value be, in addition to keep every thing else constant
* **Data Cleaning and Preprocessing**
  + **Cleaning:**
    - Determine if any columns have missing values, if there percentage of missing values is high, it would be wise to not include them in the model as it would be inaccurate and result in a bias
      * Automatically skew the data
      * Just drop it if it is missing a majority (should be comfortable dropping if missing values exceed 10%)
* **Data Exploration and Visualization**
  + **Inter-Quantile-Range[IQR] for Removing Outliers**
    - Removing upper 75% and lower 25% percentile to try to get the normal median values
    - Q3: 75% and below values
    - Q1: 25% and below values
    - IQR = Q3 – Q1 🡪 values 25%-75%
    - Remove smallest median house values and highest median house values
  + **Box-plot to remove outliers 3:12**