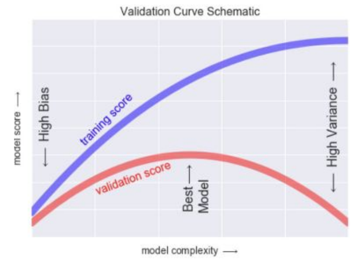
Data Input
Data Processing (Identify missing values, EDA)
ML Model

1. Load and arrange data
2. Choose model
3. Instantiate model and select hyperparameters
4. Fit model
5. Predict values for new data



Validation
- Cross-validation ,accuracy_score, R2 score, MSE
- For high-bias (underfit) models, the performance of the model on the validation set is similar to the performance on the training set
- For high-variance models (overfit), the performance of the model on the validation set is far worse than the performance on the training set

Visualization of Results

<u>Linear Model</u>

Regression: finds w and b (parameters learned) that minimize the MSE between prediction and true values, y, for training set.

*ŷ = w[0] * x[0] + … + w[p] * x[p] + b, where x is num of features*

Classification: if ŷ < zero, predict –1, otherwise, +1
- Regularization parameter, alpha (regression), and C (classification)
- If only few features are important, use L1 (Lasso), otherwise, L2 (Ridge)

Strengths:
- Fast to train and predict
- Scale to large datasets and work well with sparse data (L1)
- Easy to understand how a prediction is made
- Perform well when the number of features is large compared to samples

Weaknesses:
- Not clear why coefficients are the way they are. Particularly true if your dataset has highly correlated features
- They are also often used on very large datasets

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$ average squares errors b/w predicted and actual values

$$Recall = \frac{TP}{TP+FN} \qquad Precision = \frac{TP}{TP+FP}$$

	Positive	Negative
Positive	TP	FN
Negative	FP	TN

- Improving recall will reduce precision
- F1 score combines precision and recall

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

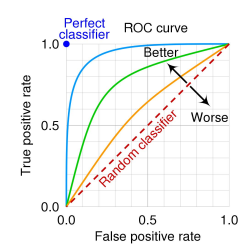
- The R2 score is the proportion variation in the target vector that can be predicted from the feature matrix
- Measure of the goodness of fit of a model
- Represents the model's ability to predict unseen samples

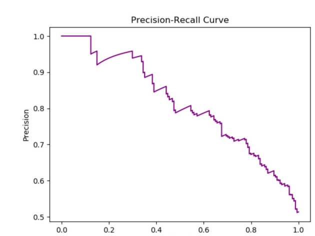
<u>Decision Function</u>
- Returns raw score indicating distance of input sample from decision boundary. Higher absolute values imply a stronger confidence in the prediction.
- values might range from -3 to +3, where negative values predict class 0, and positive values predict class 1.

<u>Prediction Probability</u>
- It returns the probability of the input sample belonging to each class. Values range from 0 to 1.
- An output might be [0.2, 0.8], indicating a 20% chance for class 0 and an 80% chance for class 1.

Precision-recall curve: Shows the tradeoff between precision and recall for different threshold

ROC curve: Shows the performance of a classification model at all classification thresholds.





Decision Tree: supervised learning
- Classification: Check which region point lies in. Traverse tree from root based on decision rules.
- Regression: Traverse tree based and find where data falls into (average target value of all training points in the leaf).

Prevent overfitting (pure leaf):
- Pre-pruning: Limit depth, number of leaves, require a min num of points
- Pruning: Remove nodes that contain little information

Feature importance: rates how important each feature is (0 and 1)

Strengths:
- Easily visualized and understood.
- No preprocessing like normalization or standardization needed.
- Algorithms invariant to scaling data. Work well when you have features on different scales, or a mix of binary and continuous.

Weaknesses:
- Tend to overfit and provide poor generalization performance

Random Forest: Collection of decision trees.
- Select num of trees (n_estimators, larger is better), inject randomness
- high max_features: similar trees, fit data easily, use distinctive features
- low max_features: different trees, might need to be deep to fit data
- Regression: make prediction, average results
- Classification: “soft” prediction, average results and predict highest class

Strength: work well w/o heavy tuning, don’t require scaling data
Weakness: time consuming, heavy resources, doesn’t perform well on high dimensional, sparse data

Gradient Boosting: Builds trees in serial manner, each tree tries to correct the mistakes of the previous one, combines simple trees
- high n_estimators means more chances to correct mistakes
- high learning_rate means making stronger correction
- GB and RF performs well on similar data

Weakness:
- Need to tune parameters, long training time
- Doesn't work well on high-dimensional sparse data

SVC Technique to add non-linear features. Learns the importance of training data to represent decision boundary between classes. Classification made on distances to support vectors.
Polynomial kernel: Computes all possible polynomials up to certain degree of the original feature (for e.g. feature1 ** 2).
Gaussian kernel (RBF): Considers all possible polynomials of all degrees, but the importance of the features decreases for higher degrees.

- C: Limits the importance of each point
- Gamma: defines how far the influence of single training example reaches, with low values (far) and high values meaning (close)

Strengths: Allow for complex decision boundaries even if data has few features. Works well for low dimensional and high dimensional data.

Weaknesses:
- Doesn't scale well with the number of samples
- Requires careful preprocessing and hyper parameter tuning
- Hard to inspect

Data Scaling (DO NOT FIT with testing data)
1. StandardScaler: Subtracts the mean (μ) and divides by the standard deviation (σ) for each feature. After scaling, mean of 0 and a sd of 1. It is sensitive to outliers.
2. RobustScaler: Uses the median (Q1) and the Interquartile Range (IQR) for scaling. This makes it robust to outliers.
3. MinMaxScaler: Subtracts min val then divides range of feature (result is [0, 1] or [a, b])
4. Normalizer: Scales the components feature vector such that the feature vector’s length is 1. Often used when direction of the data matters more than the magnitude.

Encoding
Ordinal encoding is used for ordinal (ordered) values
One hot encoding is used for nominal (unordered) values