Q What is Machine Learning
Ans: Machine Learning is about building system that can learn from data. Learning means getting better at some task, given some performance measure.

Machine Learning Types:

- **Regression:** In this algorithm learns to establish a relationship between input features and continuous output values. The goal is to predict numerical outcomes or quantities, making it useful for tasks such as price prediction, sales forecasting, and trend analysis.

- **Classification:** In this algorithm learns to categorize data into predefined classes based on labeled training examples. The goal is to predict the class of new, unseen data accurately.

Regression Techniques:

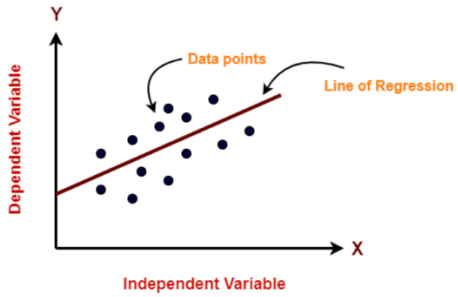
- Simple Linear Regression
- Multiple linear Regression
- Polynomial Linear Regression
- Support Vector for Regression (SVR)
- Decision Tree Regression
- Random Forest Regression

Classification Techniques:

- Logistic Regression
- Support Vector Machine SVM
- Decision Tree Classifier
- Random Forest Classifier
- K-NN K-Nearest Neighbours
- Kernel SVM
- Naive Bayes
- SGDClassifier

1. **Linear regression**: Linear regression builds a model which establishes a relationship between feature (independent variable) and target (Dependent variable).
e.g House price: Target = house price, Features = [size, location]

f(x) = wX + b → np.dot(w, X) + b [X is feature value, w is weight, b is bias]
Numpy np function uses parallel hardware to efficiently calculate the dot product.



Loss: It is a measure of the difference of a single example to its target value.
(prediction – target → y_hat - y)
Cost: It is a measure how well our model is predicting the target.

Cost Function: Squared Error Cost Function

Cost function: Squared error cost function

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(\underbrace{\hat{y}^{(i)}}_{\text{error}} - y^{(i)}\right)^2$$

m = number of training examples

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)^2$$

Gradient Decent: Gradient Descent is an iterative optimization algorithm used to minimize the loss function in machine learning models. It aims to find the optimal set of parameters (weights and biases) that result in the best performance of the model on the training data.

Repeat until convergence.

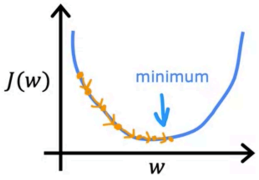
Correct: Simultaneous update

$$tmp\_w = w - \alpha \frac{\partial}{\partial w} J(w,b)$$
$$tmp\_b = b - \alpha \frac{\partial}{\partial b} J(w,b)$$
$$w = tmp\_w$$
$$b = tmp\_b$$

here alpha is learning rate.

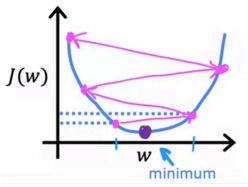
The learning rate is a crucial hyperparameter in Gradient Descent.

If alpha/learning-rate is too small, Gradient Decent may be slow.



If alpha/learning-rate is too large, Gradient Decent may

- overshoot, never reach minimum.
- Fail to converge, diverge



There are variations of Gradient Descent, such as Stochastic Gradient Descent (SGD), Mini-batch Gradient Descent, and Adam (Adaptive Moment Estimation), which use different strategies to update the parameters and address various issues like computational efficiency and adaptability to varying learning rates.

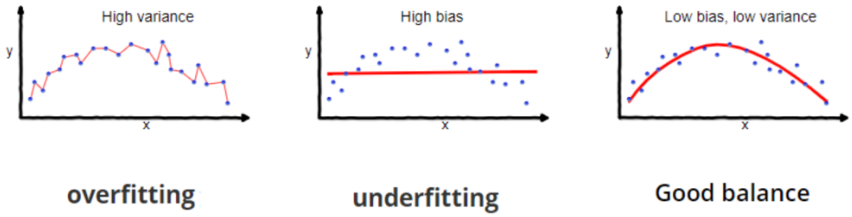
What is cross-validation?
Cross-validation is a technique used to assess a model's performance and generalization ability. The data is split into multiple subsets (folds), and the model is trained and evaluated on different combinations of these subsets to get a more reliable performance estimate.

Bias & Variance Trade off
What is Bias (Underfitting)?
Model with high bias pays very little attention to the training data and oversimplifies the model. It always leads to high error on training and test data.

- Error between average model prediction and ground truth
- The bias of the estimated function tells us the capacity of the underlying model to predict the values

What is Variance (Overfitting)?
Model with high variance pays a lot of attention to training data and does not generalise on the data which it hasn't seen before. As a result, such models perform very well on training data but has high error rates on test data.

- Average variability in the model prediction for the given dataset
- The variance of the estimated function tells you how much the function can adjust to the change in the dataset.



Why is Bias Variance Tradeoff?
If our model is too simple and has very few parameters then it may have high bias and low variance. On the other hand if our model has large number of parameters then it's going to have high variance and low bias. So we need to find the right/good balance without overfitting and underfitting the data.
Underfitting, Underfitting occurs when a machine learning model is too simple to capture the underlying patterns in the data, resulting in poor performance on both the training and test data. It often happens when the model lacks complexity or is trained on insufficient data.

How to prevent Underfitting:
- Use a more complex model with a higher capacity (e.g., increase the number of layers in a neural network).
- Increase the number of features or use more informative features.
- Train the model for a longer time or with more iterations.
- Reduce regularization to allow the model to fit the data better.
- Ensure you have enough diverse and representative training data.

Overfitting, Overfitting occurs when a machine learning model performs extremely well on the training data but poorly on unseen data. It happens when the model learns noise or specific patterns that are present only in the training data and fail to generalize to new data.

How to prevent **Overfitting**:
- To simplify the model by selecting one with fewer parameters (e.g., a linear model rather than a high-degree polynomial model), by reducing the number of attributes in the training data or by constraining the model
- Regularization (L1 and L2)
- Early Stopping
- To gather more training data
- To reduce the noise in the training data (e.g., fix data errors and remove outliers)
- Using Ensemble methods.

Q 9: What is an online learning System?
Ans: An online learning system can learn incrementally, as opposed to a batch learning system. This makes it capable of adopting rapidly to both changing data and autonomous system, and training on very large quantity of data.

Q 10: What is out-of-core learning?
Ans: Out-of-core algorithms can handle vast quantities of data that can't fit in a computer main memory. An out-of-core learning algorithm chops the data into mini-batches and uses online learning techniques to learn from these mini batches.

Q 12: What is the difference between a model parameter and a learning algorithm Hyperparameter

Ans: A model has one or more model parameters that determine what it will predict given a new instance (e.g. the slope of a linear model). A learning algorithm tries to find optimal values for these parameters such that the model generalizes well to new instances. A Hyperparameter is a parameter of a learning algorithm itself, not of the model. (e.g. the amount of regularisation is to apply).

Why do we need a Train set, Validation set and Test set?

When training a model, we divide the available data into three separate sets:

- The training dataset is used for fitting the model's parameters. However, the accuracy that we achieve on the training set is not reliable for predicting if the model will be accurate on new samples.
- The validation dataset is used to measure how well the model does on examples that weren't part of the training dataset. The metrics computed on the validation data can be used to tune the hyperparameters of the model. However, every time we evaluate the validation data and we make decisions based on those scores, we are leaking information from the validation data into our model. The more evaluations, the more information is leaked. So we can end up overfitting to the validation data, and once again the validation score won't be reliable for predicting the behaviour of the model in the real world.
- The test dataset is used to measure how well the model does on previously unseen examples. It should only be used once we have tuned the parameters using the validation set.

So if we omit the test set and only use a validation set, the validation score won't be a good estimate of the generalization of the model.

Q: What is an imbalanced dataset? Can you list some ways to deal with it?

Ans: An imbalanced dataset is one that has different proportions of target categories. For example, a dataset with medical images where we have to detect some illness will typically have many more negative samples than positive samples—say, 98% of images are without the illness and 2% of images are with the illness.

There are different options to deal with imbalanced datasets:

- Oversampling or undersampling. Instead of sampling with a uniform distribution from the training dataset, we can use other distributions so the model sees a more balanced dataset.

- Data augmentation. We can add data in the less frequent categories by modifying existing data in a controlled way. In the example dataset, we could flip the images with illnesses, or add noise to copies of the images in such a way that the illness remains visible.

- Using appropriate metrics. In the example dataset, if we had a model that always made negative predictions, it would achieve a precision of 98%. There are other metrics such as precision, recall, and F1-score that describe the accuracy of the model better when using an imbalanced dataset.

Fine Tune Your Model / Hyperparameter Tuning:

Hyperparameter tuning, also known as hyperparameter optimization, is the process of finding the best combination of hyperparameters for a machine learning model. Hyperparameters are

configuration settings that are not learned during the training process but are set before training begins. They significantly influence the model's performance and generalization ability.
The goal of hyperparameter tuning is to find the hyperparameter values that result in the best model performance on the validation or cross-validation set. The process typically involves trying different combinations of hyperparameter values and evaluating the model's performance using a chosen metric (e.g., accuracy, F1 score, or mean squared error).
There are several techniques for hyperparameter tuning, including:

- Grid Search: Exhaustively searching all possible combinations of hyperparameter values within specified ranges.
- Random Search: Sampling random combinations of hyperparameter values within specified ranges.
- Bayesian Optimization: An intelligent search algorithm that models the performance function and selects the next hyperparameter combination based on previous results.
- Genetic Algorithms: Using principles inspired by natural selection to evolve and improve hyperparameter combinations over successive iterations.
- Automated Machine Learning (AutoML): Utilizing automated tools or libraries that internally perform hyperparameter tuning along with other aspects of the machine learning pipeline.

Hyperparameter tuning is crucial to achieve optimal model performance and avoid issues like underfitting or overfitting. It helps fine-tune the model to better generalize and make accurate predictions on unseen data.

---------- Code ----------

```
from sklearn.model_selection import GridSearchCV
param_grid = [
        { n_estimators=[3, 10, 30], max_features=[2, 4, 6, 8] },
        { n_estimators=[3, 10], max_features=[2, 3, 4], bootstrap=False },
]

model = RandomForestRegressor()
grid_search = GridSearchCV( model,
                            param_grid,
                            cv=5,
                            scoring='negative_mean_squared_error',
                            return_train_score=True )
```

Numeric Imputer Strategy: mean, median, most_frequent

Best combination of Hyperparameters:

```
grid_search.best_params_
{ max_features=8, n_estimators=30}
```

Find best estimators:

```
estimator = grid_search.best_estimator_
```

---------- Code end ----------

RandomSearchCV is best as compare to GridSearchCV

. It picks random hyper parameter values, e.g. 1000 iterations , this approach will explore 1000 different value of each hyper parameters.
. We are easily control over the computing budget by controlling one value iteration=500

Model Performance metrics:

Regression Metrics:

- Mean Absolute Error (MAE): It calculates the average absolute difference between the predicted and actual values, providing a measure of the model's average prediction error.
- Mean Squared Error (MSE): It calculates the average squared difference between the predicted and actual values, giving higher weight to larger errors compared to MAE.
- Root Mean Squared Error (RMSE): It is the square root of MSE, providing a measure of the average prediction error with the same unit as the target variable.
- R-squared (Coefficient of Determination): It measures the proportion of the variance in the target variable that can be explained by the model. It ranges from 0 to 1, with higher values indicating a better fit.

Classification Metrics:

- **Accuracy**: It measures the overall correctness of the model's predictions by calculating the ratio of correct predictions to the total number of predictions.
- **Precision**: Exactness of Model. It quantifies the proportion of correctly predicted positive instances out of all instances predicted as positive. It is particularly useful when the goal is to minimize false positives e.g (Health care, medical screening, drug testing).
- **Recall** (Sensitivity or True Positive Rate): Completeness of Model. It measures the proportion of actual positive instances correctly predicted by the model. It is useful when the goal is to minimize false negatives e.g. (Fraud Detection).
- **F1-Score**: It combines precision and recall into a single metric, providing a balanced measure of the model's performance. It is the harmonic mean of precision and recall.
- **Specificity** (True Negative Rate): It measures the proportion of actual negative instances correctly predicted by the model. It is particularly useful when the goal is to minimize false negatives.

Confusion Metric: Row represents actual classes, Column represents predictions

TP 53057, FP 1522 FN 1325, TN 4096	Precision : TP / (TP + FP) —> 53057 / (53057 + 1522) Recall: TP / (TP + FN) —> 53057 / (53057+1325) F1 Score: TP / (TP + (FN + FP) / 2)