INTRODUCTION TO MACHINE LEARNING ALGORITHMS





OTDMUG, SPRING 2025

- Machine Learning Applications (pg. 3)
- Traditional Programming vs Machine Learning (pg. 7)
- Supervised vs Unsupervised Learning (pg. 8)
- Machine Learning Algorithms (pg. 9)
- ScikitLearn and DataCamp Cheat Sheet (pg. 10)
- Supervised Algorithms (pg. 13)
- Unsupervised Algorithms (pg. 36)



MACHINE LEARNING APPLICATIONS



Source: Applications of Machine learning - SwissCognitive - The Global AI Hub



MACHINE LEARNING AT USPS



The United States Postal Service

Artificial intelligence technology enables efficiency Venu Govindaraju's work in handwriting recognition was at the center of the first handwritten address interpretation system used by the United States Postal Service (USPS). USPS issued a contract to researchers at the University at Buffalo to develop the handwriting recognition technology. One year after implementation it saved the USPS ^{\$}90 million by automatically processing, and barcoding for precise delivery, more than 25 billion letters. The 2009 Computing Community Consortium dubbed the project as "one of the most successful applications of Machine Learning for developing a real-time engineered system."

This project started in 1996

Source: <u>Case Study - AI at USPS - Vice President Research and Economic Development - University at Buffalo</u> USPS video presentation: <u>Systems at Work</u>



MACHINE LEARNING AT USPS





USPS video presentation: Systems at Work



MACHINE LEARNING AT USPS



- Thousands of scanning devices and cameras used in processing packages.
- Processed more than 129 billion pieces of mail and about 7.3 billion packages in 2020.
- Generated more than **20TB of image data every day** in 2020.

"They do image <u>classification</u> and <u>object detection</u> on their packages"

Source: How the USPS Is Using AI at the Edge to Improve Mail



TRADITIONAL PROGRAMMING VS MACHINE LEARNING

Traditional Programming





SUPERVISED VS UNSUPERVISED

Supervised Learning

- Labeled data (output is known)
- Fit models that map inputs and outputs
- Make predictions on unseen data
- \circ Regression and Classification

Unsupervised Learning

- Unlabeled data (output is unknown)
- Discover hidden patterns within the data
- Recognize groups/structures
- Clustering, Dimensionality
 Reduction, Feature Extraction,
 Association, Recomendation



MACHINE LEARNING ALGORITHMS





SKLEARN CHEAT SHEET



Source: 12. Choosing the right estimator — scikit-learn 1.6.1 documentation



DATACAMP CHEAT SHEET

			ALGORITHM	DESCRIPTION	APPLICATIONS	ADVANTAGES	DISADVANTAGES
			Linear Regression	A simple algorithm that models a linear relationship between inputs and a continuous numerical output variable	USE CARES 1. Stock price prediction 2. Predicting housing prices 3. Predicting customer lifetime value	 Explainable method Interpretable results by its output coefficients Faster to train than other machine learning models 	1. Assumes linearity between inputs and output 2. Sensitive to outliers 3. Can underfit with small, high-dimensional data
smi		Aodels	Logistic Regression	A simple algorithm that models a linear relationship between inputs and a categorical output (1 or O)	USE CASES 1. Credit risk score prediction 2. Customer churn prediction	1. Interpretable and explainable 2. Less prone to overfitting when using regularization 3. Applicable for multi-class predictions	1. Assumes linearity between inputs and outputs 2. Can overfit with small, high-dimensional data
orith		Linear N	Ridge Regression	Part of the regression family — it penalizes features that have low predictive outcomes by shrinking their coefficients closer to zero. Can be used for clossification or regression	VSE CASES 1. Predictive maintenance for automobiles 2. Sales revenue prediction	1. Less prone to overfitting 2. Best suited where data suffer from multicollinearity 3. Explainable & interpretable	1. All the predictors are kept in the final model 2. Doesn't perform feature selection
Algo			Lasso Regression	Part of the regression family — it penalizes features that have low predictive outcomes by shrinking their coefficients to zero. Can be used for classification or regression	USE CASES 1. Predicting housing prices 2. Predicting clinical outcomes based on health data	1. Less prone to overfitting 2. Can handle high-titimensional data 3. No need for feature selection	 Can lead to poor interpretability as it can keep highly correlated variables
ing ,			Decision Tree	Decision Tree models make decision rules on the features to produce predictions. It can be used for classification or regression	USE CASES 1. Customer chum prediction 2. Credit score modeling 3. Disease prediction	1. Explainable and interpretable 2. Can handle missing values	1. Prone to overfitting 2. Sensitive to outliers
arn		els	Random Forests	An ensemble learning method that combines the output of multiple decision trees	USE CASES 1. Credit score modeling 2. Predicting housing prices	1. Reduces overfitting 2. Higher accuracy compared to other models	1. Training complexity can be high 2. Not very interpretable
e Le	Learning	Based Mod	Gradient Boosting Regression	Gradient Boosting Regression employs boosting to make predictive models from an ensemble of weak predictive learners	VSE CASES 1. Predicting car emissions 2. Predicting ride halling fare amount	1. Better accuracy compared to other regression models 2. It can handle multicollinearity 3. It can handle non-linear relationships	 Sensitive to outliers and can therefore cause overfitting Computationally expensive and has high complexity
hin	pervised	Tree-	XGBoost	Gradient Boosting algorithm that is efficient & flexible. Can be used for both classification and regression tasks	use cases 1. Chum prediction 2. Cialms processing in insurance	1. Provides accurate results 2. Captures non linear relationships	1. Hyperparameter tuning can be complex 2. Does not perform well on sparse datasets
Mac	dns <		LightGBM Regressor	A gradient boosting framework that is designed to be more efficient than other implementations	USE CASES 1. Predicting flight time for pirlines 2. Predicting cholestorol levels based on health data	1. Can handle large amounts of data 2. Computational efficient & fast training speed 3. Low memory usage	 Can overfit due to leaf-wise splitting and high sensitivity Hyperparameter tuning can be complex
l qo			K-Means	K-Means is the most widely used clustering approach—it determines K clusters based on euclidean distances	USE CASES 1. Customer segmentation 2. Recommendation systems	1. Scales to large datasets 2. Simple to implement and interpret 3. Results in tight clusters	1. Requires the expected number of clusters from the beginning 2. Has troubles with varying cluster sizes and densities
tacamp T	pervised Learning	Clustering	Hierarchical Clustering	A "bottom-up" appraach where each data point is treated as its own cluster—and then the closest two clusters are merged together iteratively	vst cases 1. Froud detection 2. Document clustering based on similarity	1. There is no need to specify the number of clusters 2. The resulting dendrogram is informative	1. Dosen't always result in the best alustering 2. Not suitable for large datasets due to high complexity
			Gaussian Mixture Models	A probabilistic model for modeling normally distributed clusters within a dataset	USE CASES 1. Outcomer segmentation 2. Recommendation systems	1. Computes a probability for an observation belonging to a cluster 2. Can identify overlapping clusters 3. More accurate results compared to K-means	1. Requires complex tuning 2. Requires setting the number of expected mixture components or clusters
P dd	vunu <	Association	Apriori algorithm	Rule based approach that identifies the most frequent itemset in a given dataset where prior knowledge of frequent itemset properties is used	VSE CASES 1. Product placements 2. Recommendation engines 3. Promotion optimization	1. Results are intuitive and interprotable 2. Exhaustive approach as it finds all rules based on the confidence and support	1. Generates many uninteresting Itemsets 2. Computationally and memory intensive. 3. Results in many overlapping item sets

Source: Machine Learning Cheat Sheet | DataCamp



DATACAMP CHEAT SHEET

	Decision Tree	Decision Tree models make decision rules on the features to produce predictions. It can be used for classification or regression	use cases 1. Customer churn prediction 2. Credit score modeling 3. Disease prediction	 Explainable and interpretable Can handle missing values 	1. Prone to overfitting 2. Sensitive to outliers
s	Random Forests	An ensemble learning method that combines the output of multiple decision trees	use cases 1. Credit score modeling 2. Predicting housing prices	 Reduces overfitting Higher accuracy compared to other models 	1. Training complexity can be high 2. Not very interpretable
Based Mode	Gradient Boosting Regression	Gradient Boosting Regression employs boosting to make predictive models from an ensemble of weak predictive learners	use cases 1. Predicting car emissions 2. Predicting ride hailing fore amount	 Better accuracy compared to other regression models It can handle multicollinearity It can handle non-linear relationships 	 Sensitive to outliers and can therefore cause overfitting Computationally expensive and has high complexity
Tree-I	XGBoost	Gradient Boosting algorithm that is efficient & flexible. Can be used for both classification and regression tasks	use cases 1. Churn prediction 2. Claims processing in insurance	1. Provides accurate results 2. Captures non línear relationships	 Hyperparameter tuning can be complex Does not perform well on sparse datasets
	LightGBM Regressor	A gradient boasting framework that is designed to be more efficient than other implementations	use cases 1. Predicting flight time for airlines 2. Predicting cholesterol levels based on health data	 Can handle large amounts of data Computational efficient & fast training speed Low memory usage 	 Can overfit due to leaf-wise splitting and high sensitivity Hyperparameter tuning can be complex

Source: Machine Learning Cheat Sheet | DataCamp





1. Linear Regression

- Assumes a straight-line relationship between dependent and independent variables (MLR when multiple independent variables)
- Minimizes the errors between observed and predicted values (OLS)
- \circ Simple, easy to implement and interpret, base model for comparisons



from sklearn.linear_model import LinearRegression
reg = LinearRegression().fit(X_train, y_train) # fit/train the model
reg.predict(X_test) # predicts y on new data
reg.score_ # R-squared
reg.coef_ # estimated coefficients
reg.intercept_ # intercept term



2. Logistic Regression

- Despite its name, it is used for (binary) classification
- Uses a sigmoid function to calculate probabilities
- Maximizes the likelihood (probabilities of observing the actual outcome)
- Simple, easy to interpret, computationally efficient
- Multinomial Logistic Regression for multiple classes. Softmax function instead.



from sklearn.linear_model import LogisticRegression clf = LogisticRegression().fit(X_train, y_train) # fit/train the model # X is a vector of continuous/discrete variables # y must be discrete clf.predict(X_test) # predicts classes on new data clf.predict_proba(X_test) # predicts probabilities on new data clf.score_ # mean accuracy

 $S(x) = rac{1}{1 \perp e^{-x}}$



3. Support Vector Machines (SVM)

- <u>Regression</u> or Classification
- Uses Kernels (functions) to map (transform) data to a higher-dimensional space that makes a linear separation possible
- \circ $% \ensuremath{\mathsf{Minimizes}}$ Minimizes the errors between the walls of the $\epsilon\text{-intensive}$ tube and the predicted values



from sklearn.svm import SVR # regressor
reg = SVR(kernel="linear").fit(X_train, y_train)
kernel = "linear", "rbf", "poly", "sigmoid"
reg.predict(X_test) # predicts y on new data
reg.score_ # R-squared
reg.coef_ # estimated coefficients
reg.intercept_ # intercept term



3. Support Vector Machines (SVM)



Source: Support Vector Regression (SVR) using linear and non-linear kernels — scikit-learn 1.6.1 documentation



3. Support Vector Machines (SVM)

- Regression or <u>Classification</u>
- Maximizes the margin while minimizing classification errors
- Regularization parameter "C" controls the trade-off between margin and errors (higher C prioritizes minimizing errors)
- Hyperparameter tuning techniques (e.g., cross-validation) help to find a good C



from sklearn.svm import SVC # classifier clf = SVC(kernel="linear", C=1) .fit(X_train, y_train) # kernel = "linear", "rbf", "poly", "sigmoid" clf.predict(X_test) # predicts classes on new data clf.predict_proba(X_test) # predicts probabilities on new data clf.score_ # mean accuracy









4. Decision Trees

- Regression or **Classification** Ο
- Uses a flowchart-like structure to make \bigcirc decisions
- Calculates impurity at all nodes to Ο determine the best node sequence (Gini index for classification, MSE for regression)

Branch



4. Decision Trees

- Dataset: Weather, Time of Day, Day of Week, Is Holiday, and Accident.
- We will try to predict (classify) whether an accident will occur in times of day under certain conditions.

	Weather	Time_of_Day	Day_of_Week	ls_Holiday	Accident
0	Foggy	Morning	Weekend	0	Yes
1	Clear	Afternoon	Weekend	1	Yes
2	Foggy	Morning	Weekend	0	Yes
3	Foggy	Afternoon	Weekend	1	Yes
4	Clear	Afternoon	Weekday	0	No
5	Clear	Evening	Weekday	0	No
6	Foggy	Afternoon	Weekend	0	Yes
7	Rainy	Evening	Weekday	1	Yes
8	Foggy	Morning	Weekday	1	Yes
9	Foggy	Morning	Weekend	0	Yes

```
X = data.drop('Accident', axis=1)
y = data['Accident']
X train, X test, y train, y test = train test split(X, y, test size=0.3, random state=0)
```

```
clf = DecisionTreeClassifier(random_state=0, max_depth=4)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
y_predprob = clf.predict_proba(X_test)
y score = clf.score(X test, y test)
```

```
print(f"Score Mean Accuracy: {y_score:.3f}")
```

```
Score Mean Accuracy: 0.917
```





5. k-Nearest Neighbors

- Classification and Regression
- Classifies by finding the k closest data points (neighbors) to a new data point
- Voting system where the majority of the neighbors win
- For regression, the result will be the average of the k neighbors
- Lazy learner. No training, just computes distances (Euclidean)
- Simple, very inefficient with large datasets, curse of dimensionality





6. Boosting (AdaBoost)

- Regression and Classification
- Ensemble technique (not a model itself)
- Sequentially combines "weak" models (learners) to create a strong model
- Following models focus on correcting the errors of the previous models
- Final prediction is a weighted (performance) sum of the individual predictions
- Prone to overfitting and bias due to weighted samples

6. Boosting (AdaBoost)

- Simple Decision Tree (stump) as "weak" learner
- \circ Calculate Gini to pick 1 variable for the first stump
- The misclassified records in this learner will have their weights increased
- Resample the dataset using weights (more of the misclassified records)
- Calculate Gini to pick...
- \circ Final prediction will be weighted





6. Boosting (AdaBoost)

• Need to pay attention to overfitting

```
clf = AdaBoostClassifier(random_state=0, n_estimators=50)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
y_predprob = clf.predict_proba(X_test)
y_score = clf.score(X_test, y_test)
```

print(f"Score Mean Accuracy: {y_score:.3f}")

Score Mean Accuracy: 1.000





6. Boosting (AdaBoost)

• Need to pay attention to overfitting







7. Bagging (Random Forest)

- Regression and Classification
- Ensemble technique (not a model itself)
- o Considers learners independent and use them in parallel
- Random sampling + Random feature selection
- Final prediction is a simple voting system (class) or average (reg)
- \circ Reduces overfitting and bias



7. Bagging (Random Forest)

- Trees are full decision trees (not stumps as in boosting)
- Each tree is trained on a subset of the dataset (random sampling)
- Each tree has a random set of features (# features is a hyperparameter)





7. Bagging (Random Forest)

clf = RandomForestClassifier(random_state=0, n_estimators=100, max_depth=None, max_features='sqrt')
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
y_predprob = clf.predict_proba(X_test)
y_score = clf.score(X_test, y_test)

```
print(f"Score Mean Accuracy: {y_score:.3f}")
```

Score Mean Accuracy: 0.983



8. Artificial Neural Network (ANN)

- Mimics how our brains work (node/neuron layers)
- \circ Data is passed forward (feed forward) among nodes
- Uses Linear Regression (!) model at each node, but it introduces non-linearity with activation functions
- Each node output = f(g(x1*w1+x2*w2+...+xn*wn) + bias)
- $_{\odot}$ $\,$ Weights and biases are initialized randomly $\,$
- Activation function f(x) transforms the data
 (ReLu, Sigmoid, tanh)
- There are no rules for sizing the network (test!)



Hidden Layers

Input Layer

Output Laver

8. Artificial Neural Network (ANN)







9. Convolutional Neural Network (CNN)

- \circ CNNs are designed to process grid-like data as images
- \circ ~ Very powerful for object detection, image classification, and image segmenting
- The hidden layers are now called convolutional layers (convolution + activation + pooling)
- Convolution = sliding filters. Filters are randomly generated kernels (dot-product functions) that highlight unique characteristics

in images (edges, texture, complex shapes)

 Pooling is another type of kernel but to reduce spatial dimensions (image size) while keeping the most important characteristics



9. Convolutional Neural Network (CNN)



Source: Convolutional Neural Networks - Basics · Machine Learning Notebook



9. Convolutional Neural Network (CNN)

Dynamic illustration of CNNs: <u>CNN Explainer: Learn CNN in your browser!</u>







10.KMeans

- Clustering (we don't know the classes)
- Groups data points into clusters (similarity)
- o k is the number of clusters and is a hyperparameter
- $_{\odot}$ $\,$ Minimizes distance between data points and cluster centers





10.KMeans

- Clustering (we don't know the classes)
- Groups data points into clusters (similarity)
- o k is the number of clusters and is a hyperparameter
- $_{\odot}$ $\,$ Minimizes distance between data points and cluster centers







