Machine Learning Research

Machine learning algorithms are programming procedures. They are methods created to solve a problem or complete a task.

Four primary machine learning algorithms:

1. Supervised learning

- Training: Uses labelled data which is data tagged with a label so an algorithm can successfully learn from it.
- Purpose: Helps the algorithm classify data as desired.

2. Unsupervised learning

- Training: Uses unlabelled data.
- Purpose: Algorithm finds patterns and creates data clusters.

3. Semi-supervised learning

- Training: Combines labelled and unlabelled data.
- Purpose: Algorithms are trained first with a small amount of labelled data, then with more unlabelled data.

4. Reinforcement learning

- Training: Assigns positive and negative values to desired and undesired actions
- Purpose: Encourages programs to maximize rewards through trial and error and avoid the negative training examples.

Machine learning models are the output of these procedures, containing the data and the procedural guidelines for using that data to predict new data.

Machine learning models

Most common machine learning problems are classification and prediction; these problems can be solved with classification or regression algorithms. The same algorithm can occasionally be useful for classification or regression models, depending on its training.

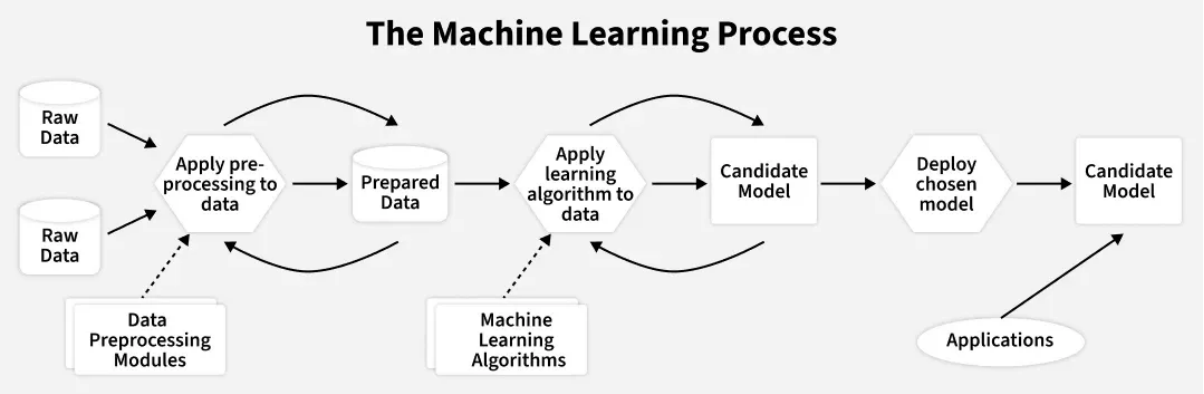
Popular algorithms for classification models.

- Logistic regression
- Naive Bayes
- Decision trees

- Random forest
- K-nearest neighbour (KNN)
- Support vector machine

Popular algorithms for regression models.

- Linear regression
- Ridge regression
- Decision trees
- Random forest
- K-nearest neighbour (KNN)
- Neural network regression



Hyperparameters

Hyperparameters are external configuration variables set manually by data scientists before training a machine learning model to govern the learning process. After the training process, parameters are formed in response to the training data, and final model parameters fits the data set perfectly.

Building process of ML model:

1. Define the Problem: Identify the type of user action, use clustering technique to divide data into logical groups.
2. Collect Data: PCAP file is uploaded by the user, that is then converted into CSV file.
3. Data Cleaning and Preprocessing: Prepare the raw data for modelling by fixing errors, handling missing values and transforming it into a clean and usable form.
4. Exploratory Data Analysis (EDA): Analyse the dataset to understand patterns, distributions and relationships between features before building the model.
5. Feature Selection and Engineering: Select the most relevant features and create new meaningful ones to enhance model accuracy, reduce complexity and improve learning.

6. Split Data into Training and Testing Sets: Divide the dataset so the model learns from one part and is evaluated fairly on unseen data.
7. Select a Machine Learning Algorithm: Choose the most suitable algorithm based on the problem type, data characteristics, and performance requirements. E.g. Centroid-based clustering: The centroid of a cluster is the arithmetic mean of all the points in the cluster. Centroid-based clustering organizes the data into non-hierarchical clusters.
8. Train the Model: Train the selected algorithm on the prepared training data so it can learn patterns, relationships and decision boundaries.
9. Evaluate Model Performance: Test the trained model on unseen data to measure how well it generalizes and how reliable its predictions are.
10. Hyperparameter Tuning: Optimize the model's hyperparameters to achieve higher accuracy, stability and better generalization.

Step-By-Step Implementation

Step 1: Import Required Libraries

- import Pandas and NumPy for data handling and numerical computations.
- import Matplotlib is used for visual analysis. associations() is used for categorical correlation.
- import pandas as pd

Step 2: Load the Dataset

```
df = pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn.csv')

df.head()
print(df.describe())
print(df.isnull().sum())
print(df['Churn'].value_counts())
```

Step 3: Data Cleaning & Feature Engineering

```
df['TotalCharges'] = df['TotalCharges'].replace('', np.nan)
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce').fillna(0)

df['SeniorCitizen'] = df['SeniorCitizen'].astype(str)

df['ChurnTarget'] = df['Churn'].apply(lambda x: 1 if x=='Yes' else 0)
```

Step 4: Feature Selection Using Correlation

```
target = 'ChurnTarget'
num_features = df.select_dtypes(include=[np.number]).columns.drop(target)
correlations = df[num_features].corrwith(df[target])
selected_num_features = correlations[abs(correlations) > 0.3].index.tolist()

cat_features = df.drop('customerID', axis=1).select_dtypes(include='object').columns
assoc = associations(df[cat_features], nominal_columns='all', plot=False)
corr_matrix = assoc['corr']

selected_cat_features = corr_matrix[corr_matrix.loc['Churn'] > 0.3].index.tolist()
selected_cat_features = selected_cat_features[:-1]

print("Selected Numeric Features:", selected_num_features)
print("Selected Categorical Features:", selected_cat_features)

selected_features = selected_num_features + selected_cat_features
```

Step 5: Train-Validation-Test Split

```
from sklearn.model_selection import train_test_split

X = df[selected_features]
y = df[target]

cat_features = X.select_dtypes(include=['object']).columns.tolist()
num_features = X.select_dtypes(include=['number']).columns.tolist()

X_train_val, X_test, y_train_val, y_test = train_test_split(X, y, test_size=0.2,
                                                            random_state=42, stratify=y)

X_train, X_val, y_train, y_val = train_test_split(X_train_val, y_train_val,
                                                  test_size=0.25, random_state=42,
                                                  stratify=y_train_val)
```

Step 6: Build ML Pipeline & Train Logistic Regression

```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

preprocessor = ColumnTransformer(
    transformers=[
        ('num', 'passthrough', num_features),
        ('cat', OneHotEncoder(handle_unknown='ignore'), cat_features)
    ])

pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(max_iter=1000))
])

pipeline.fit(X_train, y_train)

y_val_pred = pipeline.predict(X_val)
print("Validation Classification Report:\n", classification_report(y_val, y_val_pred))
```

Step 7: Hyperparameter Tuning Using GridSearchCV

```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'classifier__C': [0.1, 1, 10, 100],
    'classifier__solver': ['lbfgs', 'liblinear']
}

grid_search = GridSearchCV(pipeline, param_grid, cv=5, scoring='recall')
grid_search.fit(X_train, y_train)

print("Best Hyperparameters:", grid_search.best_params_)

y_test_pred = grid_search.predict(X_test)
print("Test Classification Report:\n", classification_report(y_test, y_test_pred))
```

Step 8: Compare Performance with Multiple ML Models

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from sklearn.metrics import recall_score

models = {
    'Logistic Regression': LogisticRegression(max_iter=1000),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(),
    'SVM': SVC(),
    'Gradient Boosting': GradientBoostingClassifier(),
    'XGBoost': XGBClassifier(use_label_encoder=False, eval_metric='logloss'),
    'LightGBM': LGBMClassifier()
}

results = []

for name, model in models.items():
    pipe = Pipeline(steps=[
        ('preprocessor', preprocessor),
        ('classifier', model)
    ])
    pipe.fit(X_train, y_train)
    y_pred = pipe.predict(X_test)
    recall = recall_score(y_test, y_pred)

    results.append((name, recall))

results
```

Step 9: Plot Recall Comparison

```
model_names = [r[0] for r in results]
test_recalls = [r[1] for r in results]

plt.figure(figsize=(10, 6))
plt.barh(model_names, test_recalls)
plt.xlabel('Recall Score')
plt.title('Model Comparison')
plt.show()
```

Step 10: Save Best Model

```
import joblib

best_model = grid_search.best_estimator_
joblib.dump(best_model, 'model.joblib')
```

References:

- Coursera. What Are Machine Learning Models and How Do You Build Them? 2025. [Online]. Available at: <machine-learning-models-link>
- Geeksforgeeks. Steps to Build a Machine Learning Model. 2025. [Online]. Available at: <https://www.geeksforgeeks.org/machine-learning/steps-to-build-a-machine-learning-model/>
- GoogleforDevelopers. Clustering Algorithms. 2025. [Online]. Available at: <https://developers.google.com/machine-learning/clustering/clustering-algorithms>