Machine Learning Training & Prediction Guide

1. Problem Understanding

Clearly define the objective: classification, regression, clustering, etc.

Example: Predict house prices (regression) or detect spam emails (classification).

2. Data Collection

Gather data from sources: CSV, databases, APIs, sensors, etc.

Tools: pandas, requests, SQL, scraping tools.

3. Data Preprocessing

Handle missing values (mean, drop, fill)

Convert categorical to numeric (LabelEncoder, OneHotEncoder)

Normalize or standardize features

Remove duplicates, irrelevant columns

Example code:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)
```

4. Exploratory Data Analysis (EDA)

Visualize data using matplotlib, seaborn

Understand correlations, distributions, outliers

Identify patterns and relationships

5. Split Data

Divide dataset into training and testing (and sometimes validation) sets

Example code:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

6. Choose a Model

Select appropriate model:

Classification: LogisticRegression, RandomForestClassifier

Regression: LinearRegression, XGBoostRegressor

Clustering: KMeans, DBSCAN

7. Train the Model

Fit the model to the training data

Example code:

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()

model.fit(X_train, y_train)
```

8. Evaluate the Model

Use the test data to evaluate the models performance

Metrics:

Classification: accuracy, precision, recall, f1-score

Regression: MAE, MSE, R

Example code:

```
from sklearn.metrics import accuracy_score

y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
```

9. Hyperparameter Tuning

Use GridSearchCV or RandomizedSearchCV to find best parameters

Example code:

```
from sklearn.model_selection import GridSearchCV

params = {'C': [0.1, 1, 10]}

grid = GridSearchCV(LogisticRegression(), params)

grid.fit(X_train, y_train)
```

10. Make Predictions

Use the trained model to make predictions on new/unseen data

Example code:

```
new_prediction = model.predict([[5.1, 3.5, 1.4, 0.2]])  # Example input
```

11. Save and Load Model

Save model for future use using joblib or pickle

Example code:

```
import joblib

joblib.dump(model, "model.pkl")

model = joblib.load("model.pkl")
```

12. Monitor & Retrain

Monitor model performance over time

Retrain with new data as patterns evolve