# Week 00b Intermediate: Supervised Learning Implementation

Machine Learning for Smarter Innovation

## 1 Week 00b Intermediate: Supervised Learning Implementation

### 1.1 Regression Example: House Price Prediction

```
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
import pandas as pd

# Load data (example)
from sklearn.datasets import fetch_california_housing
housing = fetch_california_housing()
X, y = housing.data, housing.target

# Split
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,
    random_state=42)

# Compare models
models = {
    'Linear': LinearRegression(),
    'Ridge': Ridge(alpha=1.0),
    'Lasso': Lasso(alpha=0.1),
    'RandomForest': RandomForestRegressor(n_estimators=100),
    'GradientBoosting': GradientBoostingRegressor(n_estimators=100)
}

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    r2 = r2_score(y_test, y_pred)
    print(f"{name}: RMSE={rmse:.3f}, R  ={r2:.3f}")
```

### 1.2 Classification: Binary and Multi-class

```
from sklearn.datasets import load_breast_cancer, load_iris
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix

# Binary classification
cancer = load_breast_cancer()
X, y = cancer.data, cancer.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

clf = LogisticRegression(max_iter=10000)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

print(classification_report(y_test, y_pred, target_names=cancer.target_names))
```

## 1.3 Hyperparameter Tuning

```
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV

param_grid = {
    ’n_estimators’: [50, 100, 200],
    ’max_depth’: [5, 10, 20, None],
    ’min_samples_split’: [2, 5, 10]
}

rf = RandomForestClassifier()
grid = GridSearchCV(rf, param_grid, cv=5, scoring=’accuracy’, n_jobs=-1)
grid.fit(X_train, y_train)

print(f"Best params: {grid.best_params_}")
print(f"Best CV score: {grid.best_score_:.3f}")
```

## 1.4 Feature Engineering

```
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.pipeline import Pipeline

# Polynomial features
pipeline = Pipeline([
    (’poly’, PolynomialFeatures(degree=2)),
    (’scaler’, StandardScaler()),
    (’model’, Ridge(alpha=1.0))
])

pipeline.fit(X_train, y_train)
score = pipeline.score(X_test, y_test)
```

## 1.5 Practice Exercises

1. Load Titanic dataset, predict survival
2. Compare logistic vs SVM on iris dataset
3. Build ensemble of 3 different models
4. Tune XGBoost hyperparameters with RandomizedSearchCV