Handout 1: Basic Classification - Your First Steps in Machine Learning

Machine Learning for Smarter Innovation

1 Handout 1: Basic Classification - Your First Steps in Machine Learning

1.1 Learning Objectives

By the end of this handout, you will: - Understand what classification is and why it matters - Build your first classifier using Python - Evaluate how well your classifier works - Apply classification to a real innovation dataset

1.2 What is Classification?

Classification is teaching a computer to sort things into groups. Just like you sort: - Emails into spam or not spam - Photos into categories (vacation, family, work) - Music into genres (rock, pop, classical)

In our innovation context, we classify: - Startups as likely to succeed or fail - Products as worth launching or not - Ideas as innovative or conventional

1.3 Your First Classifier in Python

1.3.1 Step 1: Import the Tools

```
# These are the basic tools we need
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import pandas as pd
```

1.3.2 Step 2: Load Data

```
# Load our innovation dataset
data = pd.read_csv('innovation_data.csv')

# Look at the first few rows
print(data.head())
```

1.3.3 Step 3: Prepare the Data

```
# X = features (what we know about each innovation)
# y = target (what we want to predict - success or failure)

X = data[['novelty_score', 'market_size', 'team_experience']]
y = data['success']  # This is 1 for success, 0 for failure
```

1.3.4 Step 4: Split into Training and Test Sets

```
# We use 80% of data to train, 20% to test
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

print(f"Training on {len(X_train)} examples")
print(f"Testing on {len(X_test)} examples")
```

1.3.5 Step 5: Train Your First Classifier

```
# Create a simple classifier (Logistic Regression)
classifier = LogisticRegression()

# Train it on the training data
classifier.fit(X_train, y_train)

print("Training complete!")
```

1.3.6 Step 6: Make Predictions

```
# Predict on the test set
predictions = classifier.predict(X_test)

# Look at first 5 predictions vs actual
for i in range(5):
    print(f"Predicted: {predictions[i]}, Actual: {y_test.iloc[i]}")
```

1.3.7 Step 7: Evaluate Performance

```
# Calculate accuracy (% correct)
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy * 100:.1f}%")

# This means our classifier got X% of predictions correct!
```

1.4 Understanding Different Classifiers

1.4.1 1. Logistic Regression - The Simple Scorer

Think of it as giving points for good features:

```
from sklearn.linear_model import LogisticRegression
```

```
model = LogisticRegression()
model.fit(X_train, y_train)
```

- Fast and simple
- Good baseline
- Easy to understand

1.4.2 2. Decision Tree - The Question Asker

Like playing 20 questions:

```
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier(max_depth=5)
model.fit(X_train, y_train)
```

- Easy to visualize
- Can explain decisions
- Might overfit if too deep

1.4.3 3. Random Forest - The Team of Experts

Many trees voting together:

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators=100)
model.fit(X_train, y_train)
```

- More accurate
- Handles complex patterns
- Harder to explain

1.5 Practical Exercise: Innovation Success Predictor

```
# Complete working example
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

# 1. Load and explore data
data = pd.read_csv('innovation_data.csv')
print(f"Dataset has {len(data)} innovations")
print(f"Success rate: {data['success'].mean() * 100:.1f}%")

# 2. Select features
feature_columns = [
    'novelty_score',       # 0-100 score
    'market_size',         # in millions
    'team_experience',     # in years
    'development_time',    # in months
    'budget_used'          # percentage of budget
]
X = data[feature_columns]
```

```
y = data['success']

# 3. Split data
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# 4. Train classifier
classifier = RandomForestClassifier(n_estimators=100, random_state=42)
classifier.fit(X_train, y_train)

# 5. Make predictions
predictions = classifier.predict(X_test)

# 6. Evaluate
accuracy = accuracy_score(y_test, predictions)
print(f"\nModel accuracy: {accuracy * 100:.1f}%")

# 7. Show feature importance
importances = classifier.feature_importances_
for feature, importance in zip(feature_columns, importances):
    print(f"{feature}: {importance * 100:.1f}% important")
```

1.6 Common Mistakes to Avoid

1. **Testing on training data** - Always keep test data separate
2. **Ignoring imbalanced data** - If 95% fail, predicting "all fail" gets 95% accuracy!
3. **Not checking feature importance** - Some features might not help at all
4. **Overfitting** - Making the model too complex for the data

1.7 Key Terms Summary

- **Classification**: Sorting into categories
- **Features**: The information we use to make decisions
- **Target**: What we're trying to predict
- **Training**: Teaching the model with examples
- **Testing**: Checking if the model learned correctly
- **Accuracy**: Percentage of correct predictions
- **Overfitting**: Memorizing instead of learning patterns

1.8 Next Steps

1. Try different classifiers on the same data
2. Add more features and see if accuracy improves
3. Visualize your decision tree to understand its logic
4. Experiment with different train/test splits

1.9 Quick Reference

```
# Import everything you need
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
import pandas as pd
```

```
# Basic workflow
data = pd.read_csv('your_data.csv')
X = data[['feature1', 'feature2', 'feature3']]
y = data['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
model = RandomForestClassifier()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy * 100:.1f}%")
```

Remember: Everyone starts somewhere. Your first classifier might only be 70% accurate, and that's okay! The goal is to understand the process and improve from there.