Day58_Overfitting_Techniques

August 5, 2025

Overfitting in Machine Learning: Concepts, Techniques & Python Code

What is Overfitting?

> Overfitting happens when a model learns not only the actual patterns in training data but also the **noise**, making it perform very well on training data but **poorly on new data** (test data). It's like memorizing answers instead of understanding the concept. - High accuracy on training data - Low accuracy on test data

If a student memorizes answers instead of understanding concepts, they may score well in practice but fail in the final exam.

Techniques to Prevent Overfitting:

1 **Cross-Validation (CV)**

- **K-Fold CV**: Split data into k parts, train on k−1 and test on the remaining.
- **Stratified K-Fold**: Keeps class distribution same across all folds.
- **GridSearchCV**: Exhaustive search over parameter combinations.
- **RandomizedSearchCV**: Random sampling of parameter combinations.

2 **Regularization**

- **Lasso (L1)**: Shrinks some coefficients to zero — feature selection.
- **Ridge (L2)**: Shrinks coefficients — helps generalization.

3 **Dimensionality Reduction**

- **PCA**: Removes less important features — reduces overfitting risk.

4 **Ensemble Models**

- **Bagging**: Combines multiple weak models — Random Forest.
- **Boosting**: Sequential improvement — Gradient Boosting.

Step-by-Step in Python (Classification Example)

[25]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Load classification data
df = pd.read_csv(r"C:\Users\Lenovo\Downloads\logit classification.csv")  #␣
 ↪Adjust path if needed
```

```
X = df[["Age", "EstimatedSalary"]].values
y = df["Purchased"].values

# Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

# Scale
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

1 Cross Validation

[27]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score, StratifiedKFold,␣
 ↪GridSearchCV, RandomizedSearchCV
from scipy.stats import uniform
```

1.1 Basic K-Fold CV

[28]:
```
# Basic K-Fold CV
scores = cross_val_score(LogisticRegression(), X_train_scaled, y_train, cv=5)
print("K-Fold Accuracy:", scores.mean())
```

```
K-Fold Accuracy: 0.8178571428571428
```

1.2 Stratified CV

[29]:
```
# Stratified CV
skf = StratifiedKFold(n_splits=5)
strat_scores = cross_val_score(LogisticRegression(), X_train_scaled, y_train,␣
 ↪cv=skf)
print("Stratified Accuracy:", strat_scores.mean())
```

```
Stratified Accuracy: 0.8178571428571428
```

1.3 GridSearchCV

[30]:
```
grid = GridSearchCV(LogisticRegression(), {'C': [0.01, 0.1, 1, 10]}, cv=5)
grid.fit(X_train_scaled, y_train)
print("GridSearch Best Params:", grid.best_params_)
```

```
GridSearch Best Params: {'C': 10}
```

1.4 RandomizedSearchCV

[31]:
```
rand = RandomizedSearchCV(LogisticRegression(), {'C': uniform(0.01, 10)},␣
↪n_iter=5, cv=5, random_state=42)
rand.fit(X_train_scaled, y_train)
print("RandomSearch Best Params:", rand.best_params_)
```

```
RandomSearch Best Params: {'C': np.float64(3.7554011884736247)}
```

2 Regularization

[32]:
```
from sklearn.linear_model import RidgeClassifier, LogisticRegressionCV

# Ridge Classifier
ridge = RidgeClassifier(alpha=1.0)
ridge.fit(X_train_scaled, y_train)
print("Ridge Score:", ridge.score(X_test_scaled, y_test))
```

```
Ridge Score: 0.85
```

[33]:
```
# Logistic with L1 or L2 via LogisticRegressionCV
logcv = LogisticRegressionCV(cv=5, penalty='l2', solver='lbfgs')
logcv.fit(X_train_scaled, y_train)
print("LogisticRegressionCV Score:", logcv.score(X_test_scaled, y_test))
```

```
LogisticRegressionCV Score: 0.85
```

3 PCA (Principal Component Analysis)

[34]:
```
from sklearn.decomposition import PCA

pca = PCA(n_components=1)
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)

pca_model = LogisticRegression()
pca_model.fit(X_train_pca, y_train)
print("PCA Model Score:", pca_model.score(X_test_pca, y_test))
```

```
PCA Model Score: 0.825
```

4 Ensemble Methods

[35]:
```
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

rf = RandomForestClassifier(n_estimators=100, max_depth=5)
rf.fit(X_train, y_train)
```

```
print("Random Forest Score:", rf.score(X_test, y_test))

gb = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1,␣
  ↪max_depth=3)
gb.fit(X_train, y_train)
print("Gradient Boosting Score:", gb.score(X_test, y_test))
```

```
Random Forest Score: 0.9416666666666667
Gradient Boosting Score: 0.9
```

Summary Table

Technique	Prevents Overfitting by
Cross Validation	Reliable evaluation (reduces variance)
Regularization	Shrinks coefficients to prevent complexity
PCA	Removes irrelevant features
Random Forest / Boosting	Uses many weak learners

[ ]: