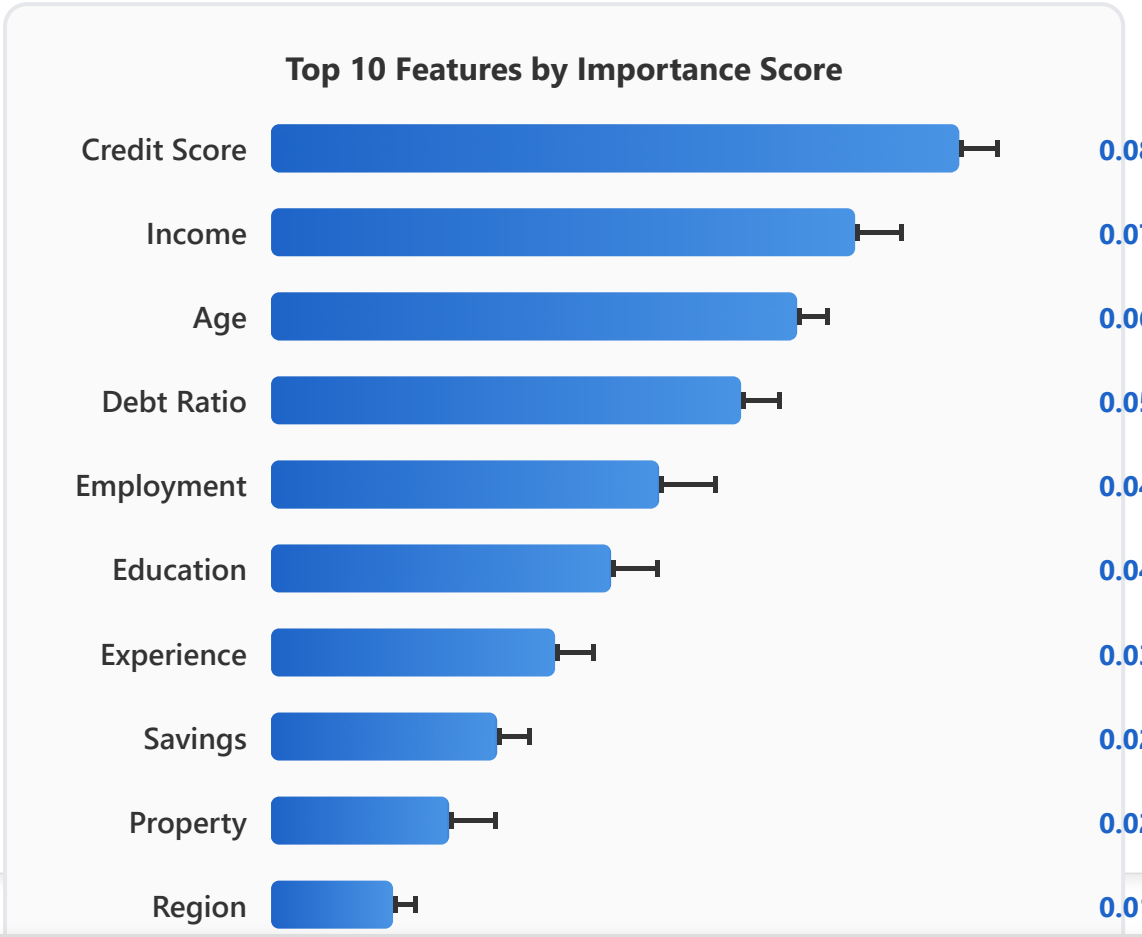
Permutation Importance

Feature Importance via Random Shuffling



How It Works

Process

1. Measure baseline performance
2. Shuffle feature values randomly
3. Re-evaluate model performance
4. Importance = Performance drop

Model-Agnostic

Works with any trained model

Interactions

Captures feature contributions including interactions

Implementation

sklearn.inspection.permutation_importance()

✓ Advantages

Simple, reliable, no assumptions

⚠ Limitations

Correlated features may deflate

Permutation Importance: Detailed Algorithm (Example)

1 Calculate Baseline Score

Evaluate the trained model on validation dataset to establish baseline performance metric

2 Select Feature to Test

Choose one feature column from the dataset to permute

```
baseline_score = model.score(X_val, y_val) →
baseline_score = 0.850 (R² score)
```

```
feature_to_test = 'Credit Score' # Testing most important feature first
```

3 Randomly Shuffle Feature

Randomly permute the values of selected feature, breaking its relationship with target

```
X_permuted = X_val.copy() X_permuted['Credit Score'] = shuffle([750, 680, 820, ...]) → [820, 750, 680, ...] # Randomly shuffled
```

4 Re-evaluate Performance

Calculate new performance score with shuffled feature values

```
permuted_score = model.score(X_permuted, y_val) →
permuted_score = 0.765 (R² dropped!)
```

5 Calculate Importance

Feature importance = Performance drop caused by shuffling

```
importance = baseline_score - permuted_score importance = 0.850 - 0.765 = 0.085 ✓
```

6 Repeat for Stability

Repeat shuffling multiple times and average for stable estimate

```
importances = [0.085, 0.082, 0.088, 0.084, 0.086]
mean_importance = 0.085 ± 0.002
```

7 Iterate All Features

Repeat process for every feature to get complete ranking

```
Credit Score: 0.085 Income: 0.072 Age: 0.065 Debt Ratio: 0.058 ...
```

8 Rank & Visualize

Sort features by importance and create visualization with error bars

```
sorted_features = sort_by_importance() → See bar chart above! 📊
```