**Ways to Detect Irrelevant Features**

Let’s go through the best techniques — from **basic statistics** to **advanced ML-based** methods.

**🔹 1. Correlation Analysis (Statistical Approach)**

**➤ For numeric features:**

You can compute correlation with the target variable.

import pandas as pd

corr = df.corr()

corr['target'].sort\_values(ascending=False)

* High positive/negative correlation → **important**
* Very low correlation (close to 0) → possibly **irrelevant**

⚠️ Note: Works only for **linear** relationships.

**🔹 2. Chi-Square Test (Categorical Features)**

Used to test whether **categorical features** are independent of the target variable.

from sklearn.feature\_selection import chi2

from sklearn.feature\_selection import SelectKBest

X\_new = SelectKBest(chi2, k=10).fit\_transform(X, y)

* If **p-value > 0.05**, the feature might be **irrelevant**.

**🔹 3. Variance Threshold (Low-Variance Filter)**

If a feature has **almost the same value** for all samples, it’s useless.

from sklearn.feature\_selection import VarianceThreshold

selector = VarianceThreshold(threshold=0.01)

X\_reduced = selector.fit\_transform(X)

* Removes features with **low information content**.

**🔹 4. Feature Importance (Model-Based Methods)**

Train a simple model and check which features matter most.

**Example: Using Random Forest**

from sklearn.ensemble import RandomForestClassifier

import pandas as pd

model = RandomForestClassifier()

model.fit(X, y)

importance = pd.Series(model.feature\_importances\_, index=X.columns)

importance.sort\_values(ascending=False)

Features with **very low importance (close to 0)** can be dropped.

🟢 Works for **nonlinear** relationships too.

**🔹 5. Recursive Feature Elimination (RFE)**

This method repeatedly builds models and removes the **least important features** each time.

from sklearn.feature\_selection import RFE

from sklearn.linear\_model import LogisticRegression

rfe = RFE(LogisticRegression(), n\_features\_to\_select=10)

rfe.fit(X, y)

selected\_features = X.columns[rfe.support\_]

* Keeps only the **most important features** automatically.

**🔹 6. Mutual Information (Nonlinear Relationship Detection)**

Measures how much information a feature gives about the target — works for both numeric and categorical data.

from sklearn.feature\_selection import mutual\_info\_classif

mi = mutual\_info\_classif(X, y)

pd.Series(mi, index=X.columns).sort\_values(ascending=False)

Features with very low mutual information → likely irrelevant.

**🔹 7. PCA (Dimensionality Reduction)**

Principal Component Analysis doesn’t tell you *which* features are irrelevant,  
but it helps identify **redundant** or **highly correlated** features.

If many features collapse into a few principal components, you know:

Several features carry overlapping information.

**🔹 8. Regularization Insight (L1 Regularization)**

If you apply **Lasso (L1)** regression, it automatically **shrinks irrelevant feature weights to zero**.

from sklearn.linear\_model import Lasso

lasso = Lasso(alpha=0.1)

lasso.fit(X, y)

pd.Series(lasso.coef\_, index=X.columns)

→ Features with **coefficient = 0** are irrelevant.

**🔹 9. SHAP or Permutation Importance (Advanced)**

For deep learning or tree models, **SHAP values** show how much each feature contributes to predictions.

import shap

explainer = shap.TreeExplainer(model)

shap\_values = explainer.shap\_values(X)

shap.summary\_plot(shap\_values, X)

Low SHAP contribution = irrelevant feature.

**🧩 Practical Workflow to Identify Irrelevant Features**

| **Step** | **Technique** | **Purpose** |
| --- | --- | --- |
| 1 | Correlation + Variance Threshold | Remove basic noise |
| 2 | Mutual Info / Chi-square | Test nonlinear importance |
| 3 | Feature Importance (Random Forest / XGBoost) | Rank all features |
| 4 | Lasso / RFE | Select top features |
| 5 | PCA (optional) | Remove redundancy among correlated features |

**📊 Example: Combined Workflow**

from sklearn.feature\_selection import VarianceThreshold, SelectKBest, mutual\_info\_classif

from sklearn.linear\_model import Lasso

# 1. Remove low variance

X1 = VarianceThreshold(0.01).fit\_transform(X)

# 2. Select features based on mutual info

mi = SelectKBest(mutual\_info\_classif, k=20).fit(X1, y)

X2 = mi.transform(X1)

# 3. Lasso for final selection

lasso = Lasso(alpha=0.01)

lasso.fit(X2, y)

selected\_features = X.columns[lasso.coef\_ != 0]

**✅ Summary**

| **Method** | **Type** | **Detects** | **Notes** |
| --- | --- | --- | --- |
| Correlation | Statistical | Linear irrelevant features | Fast but simple |
| Chi-Square | Statistical | Categorical | Needs discrete target |
| Variance Threshold | Filter | Constant/near-constant | Quick prefilter |
| Mutual Information | Filter | Nonlinear relevance | Works well generally |
| Model-Based Importance | Wrapper | Nonlinear + interactions | Good for tree models |
| L1 (Lasso) | Embedded | Zeroes out irrelevant features | Built-in feature selection |
| PCA | Dimensionality | Redundant features | Reduces feature space |