- **Feature Creation** → creates new features (is_home_team, run_margin_per_over).
- **Feature Transformation** → label encoding, one-hot encoding, polynomial features.
- **Feature Extraction** → uses **PCA** to reduce numeric features to principal components.
- **Feature Selection** → selects the most important features using **chi-square test**.
- **Feature Scaling** → applies **StandardScaler** and **MinMaxScaler** to normalize values.

📌 Final Code: Feature Engineering on IPL Dataset

python Copy code

```python
# ============================================================
# Foundations of Data Science Project
# Dataset: Cleaned_Matches_Dataset.csv
# Topic: Feature Engineering
# ============================================================

import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler, MinMaxScaler, Polyn
from sklearn.decomposition import PCA
from sklearn.feature_selection import SelectKBest, chi2
import matplotlib.pyplot as plt
```

Copy code

```python
# ================================
# 1. Load Dataset
# ================================
df = pd.read_csv("Cleaned_Matches_Dataset.csv")

# Drop unwanted index column if present
if "Unnamed: 0" in df.columns:
    df = df.drop(columns=["Unnamed: 0"])

print("Shape of dataset:", df.shape)
print(df.head())

# ============================================================
# 2. Feature Creation
# ============================================================
print("\n=== Feature Creation ===")
# Example: Create a feature "is_home_team" (whether team1 plays in its home city)
df["is_home_team"] = (df["city"] == df["team1"]).astype(int)

# Example: Create "run_margin_per_over" from result_margin / target_overs
df["run_margin_per_over"] = df["result_margin"] / df["target_overs"]
```

```
# Example: Create "run_margin_per_over" from result_margin / target_overs
df["run_margin_per_over"] = df["result_margin"] / df["target_overs"]

print(df[["team1","city","is_home_team","result_margin","target_overs","run_margin_per_over"]].hea
```

Copy code

```
# ============================================================
# 3. Feature Transformation
# ============================================================
print("\n=== Feature Transformation ===")
# Encode categorical feature "toss_decision"
df["toss_decision_encoded"] = LabelEncoder().fit_transform(df["toss_decision"].astype(str))

# One-hot encoding for "venue" (first 5 columns shown)
df_onehot = pd.get_dummies(df["venue"], prefix="venue")
print("One-hot encoded venue (sample):\n", df_onehot.head())

# Polynomial feature example (squares of numeric values)
poly = PolynomialFeatures(degree=2, include_bias=False)
num_features = df[["target_runs","target_overs"]].fillna(0)
poly_features = poly.fit_transform(num_features)
print("Polynomial features shape:", poly_features.shape)
```

Copy code

```
# ============================================================
# 4. Feature Extraction (PCA)
# ============================================================
print("\n=== Feature Extraction (PCA) ===")
# Apply PCA on numeric features
num_cols = ["result_margin","target_runs","target_overs"]
df_num = df[num_cols].fillna(0)

pca = PCA(n_components=2)
df_pca = pca.fit_transform(df_num)
print("Explained Variance Ratio:", pca.explained_variance_ratio_)

plt.scatter(df_pca[:,0], df_pca[:,1])
plt.title("PCA Feature Extraction")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.show()
```

Copy code

```
# ============================================================
# 5. Feature Selection
# ============================================================
print("\n=== Feature Selection ===")
# Select top 2 features using chi-square test
df_fs = df.dropna(subset=["result_margin","target_runs","target_overs","toss_decision_encoded"]).c
X = df_fs[["target_runs","target_overs","toss_decision_encoded"]]
y = (df_fs["result_margin"] > 20).astype(int)  # classify big win vs small win

selector = SelectKBest(chi2, k=2)
X_new = selector.fit_transform(X, y)
selected_features = X.columns[selector.get_support()]
print("Selected features:", selected_features.tolist())
```

```python
# ============================================================
# 6. Feature Scaling
# ============================================================
print("\n=== Feature Scaling ===")
scaler1 = StandardScaler()
scaler2 = MinMaxScaler()

scaled_standard = scaler1.fit_transform(df_num)
scaled_minmax = scaler2.fit_transform(df_num)

print("Original:\n", df_num.head())
print("Standard Scaled:\n", scaled_standard[:5])
print("Min-Max Scaled:\n", scaled_minmax[:5])
```