Cheat Sheet for Data Analysis #6Dimensionality Reduction (PCA & t-SNE)

Basic Setup and Data Loading

Import essential libraries

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
import seaborn as sns
```

Meaning: Imports necessary libraries for dimensionality reduction techniques

Load a CSV file into a DataFrame

```
df = pd.read_csv("Diamond.csv")
```

Meaning: Reads data from a CSV file and stores it in a variable called `df`

View basic dataset information

```
print("Dataset shape:", df.shape)
df.head()
```

Meaning: Displays the dimensions of the dataset and the first few rows for quick inspection

---

Prepare Data for Dimensionality Reduction

Identify categorical variables

```
print(df.dtypes)
```

Meaning: `.dtypes` helps identify which columns are categorical (typically 'object' type) and need encoding

One-hot encode categorical features

```
# Convert categorical variables to dummy variables
categorical_cols = ['colour', 'clarity', 'certification']
df_encoded = pd.get_dummies(df, columns=categorical_cols, drop_first=True)
print(f"Shape after one-hot encoding: {df_encoded.shape}")
```

Meaning: Transforms categorical variables into numerical format required for PCA/t-SNE; increases dimensionality but enables analysis

Standardize the data

```
scaler = StandardScaler()
X_std = scaler.fit_transform(df_encoded)
X_std_df = pd.DataFrame(X_std, columns=df_encoded.columns)
print("Standardized data (mean  0, std  1):")
print("Mean per feature:", np.round(X_std_df.mean(), 2).values[:5], "...")
print("Std per feature:", np.round(X_std_df.std(), 2).values[:5], "...")
```

Meaning: Ensures all features contribute equally to PCA by centering (mean=0) and scaling (std=1); prevents features with larger scales (like price) from dominating

---

Principal Component Analysis (PCA)

Apply PCA and examine explained variance

```
pca = PCA()
PC = pca.fit_transform(X_std)

# Explained variance
explained_var = pca.explained_variance_
explained_var_ratio = pca.explained_variance_ratio_

print("Total variance (should equal number of features):", np.sum(explained_var))
print("Variance of first 5 PCs:", np.round(explained_var[:5], 2))
print("Cumulative explained variance by first 5 PCs:",
      f"{np.sum(explained_var_ratio[:5]):.1%}")
```

Meaning: PCA identifies principal components that capture decreasing amounts of variance in the data

Plot explained variance

```
plt.figure(figsize=(10, 4))

plt.subplot(1, 2, 1)
plt.plot(range(1, len(explained_var)+1), explained_var, 'bo-')
plt.title('Variance of Each Principal Component')
plt.xlabel('Principal Component')
plt.ylabel('Variance (Eigenvalue  _j)')

plt.subplot(1, 2, 2)
plt.plot(range(1, len(explained_var)+1), np.cumsum(explained_var_ratio), 'ro-')
plt.title('Cumulative Explained Variance')
plt.xlabel('Number of Components')
```

```
plt.ylabel('Cumulative Proportion of Variance')
plt.axhline(y=0.9, color='k', linestyle='--', alpha=0.7)
plt.text(2, 0.92, '90% threshold', fontsize=10)

plt.tight_layout()
plt.show()
```

Meaning: Visualizes how much variance each component explains and helps determine how many components to retain

Determine number of components for 90% variance

```
cumsum_var = np.cumsum(explained_var_ratio)
n_components_90 = np.argmax(cumsum_var >= 0.9) + 1
print(f"Number of PCs needed for 90% variance: {n_components_90}")
```

Meaning: Finds the minimum number of components that capture at least 90% of total variance

Examine PCA loadings

```
# Loadings show how original features contribute to each PC
loadings = pca.components_.T
loading_df = pd.DataFrame(
    loadings[:, :5],
    index=df_encoded.columns,
    columns=[f'PC{i+1}' for i in range(5)]
)

print("\nLoadings for first 2 PCs (absolute values, top 5 features):")
print(loading_df[['PC1', 'PC2']].abs().sum(axis=1).sort_values(ascending=False).head())
```

Meaning: High absolute loading values indicate that an original feature strongly influences that principal component

Key Concepts in PCA

What do the principal components represent? **Principal components are orthogonal linear combinations of the standardized features.** Each PC is a new axis in the transformed space, constructed as a weighted sum of all original features, where the weights are chosen to maximize the variance captured along that axis.

What does a high eigenvalue ( ) in PCA indicate? A **high eigenvalue ( )** associated with a principal component indicates two important things: 1. The PC **explains a lot of variance** in the original dataset 2. The PC is **important for dimensionality reduction** - it captures significant information and should be retained when reducing dimensions

The eigenvalue is numerically equal to the variance explained by that component.

---

t-SNE for Nonlinear Dimensionality Reduction

Apply t-SNE for 2D visualization

```
tsne = TSNE(n_components=2, perplexity=30, random_state=42, max_iter=1000)
X_tsne = tsne.fit_transform(X_std)

# Plot t-SNE result
plt.figure(figsize=(8, 6))
plt.scatter(X_tsne[:, 0], X_tsne[:, 1], alpha=0.7)
plt.title('t-SNE Embedding of Diamond Dataset (2D)')
plt.xlabel('t-SNE 1')
plt.ylabel('t-SNE 2')
plt.show()
```

Meaning: Creates a 2D map where similar diamonds are placed close together; excellent for visualizing clusters

---

Compare PCA and t-SNE Visualizations

Side-by-side comparison

```
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.scatter(PC[:, 0], PC[:, 1], alpha=0.7)
plt.title(f'PCA (PC1 vs PC2)\nExplained variance: {explained_var_ratio[0]:.1%} + {explained_
plt.xlabel('PC1')
plt.ylabel('PC2')

plt.subplot(1, 2, 2)
plt.scatter(X_tsne[:, 0], X_tsne[:, 1], alpha=0.7)
plt.title('t-SNE (2D)')
plt.xlabel('t-SNE 1')
plt.ylabel('t-SNE 2')

plt.tight_layout()
plt.show()
```

Meaning: Compares linear (PCA) and nonlinear (t-SNE) dimensionality reduction approaches

Practical Tips for Dimensionality Reduction

1. **Always standardize data before PCA:** Features with different scales will distort results
2. **Choose the right method:**
   - Use **PCA** for dimensionality reduction before modeling (linear relationships)
   - Use **t-SNE** for exploratory visualization and cluster detection (nonlinear relationships)
3. **Interpret PCA components carefully:** Examine loadings to understand what each PC represents in terms of original features
4. **Handle high-dimensional encoded data:** After one-hot encoding, PCA can effectively reduce the expanded feature space
5. **Set appropriate perplexity in t-SNE:** Typically between 5-50; balances attention to local vs. global structure
6. **Remember t-SNE is stochastic:** Running it multiple times may yield different layouts
7. **Use cumulative variance plot:** To determine how many PCA components to retain for your analysis
8. **Preserve global vs. local structure:**
   - PCA preserves global variance and overall data structure
   - t-SNE preserves local similarities and neighborhood relationships

Remember: Dimensionality reduction transforms complex, high-dimensional data into lower-dimensional representations while preserving essential patterns. PCA provides interpretable components based on variance, while t-SNE excels at revealing hidden clusters in nonlinear data structures.