Customer Journey Analysis Using Clustering and Dimensionality Reduction: Enhancing User Experience

Phase 3: Model Training and Evaluation

3.1 Overview of Model Training and Evaluation

In this phase, we focus on selecting suitable algorithms, training the models using the processed data, and evaluating their performance. The goal is to enhance customer journey analysis by identifying key user segments and improving personalization. Various machine learning techniques, including clustering and dimensionality reduction, are employed to uncover patterns in user behavior. Hyperparameter tuning is performed to optimize model performance, and multiple evaluation metrics are used to assess predictive capabilities. Cross-validation ensures that the model generalizes well to unseen data.

3.2 Suitable Algorithms

For the **Customer Journey Analysis Using Clustering and Dimensionality Reduction** project, we use the following algorithms:

1. **Dimensionality Reduction (Feature Extraction):**
   - **Principal Component Analysis (PCA):** Reduces the dataset's dimensionality while preserving variance.
   - **Autoencoder (Neural Network-based Feature Extraction):** Encodes customer behavior data into a lower-dimensional latent space.
2. **Clustering (Customer Segmentation):**
   - **K-Means Clustering:** Groups customers into distinct segments based on behavioral similarities.
   - **Hierarchical Clustering:** Creates a hierarchy of customer groups, allowing dynamic segmentation.
3. **Classification & Prediction Models:**
   - **Decision Trees:** Provides interpretable segmentation based on user attributes.
   - **Neural Networks:** Captures complex relationships in customer behavior for personal recommendations.

Source code:

```
# Based on the Elbow Method, optimal K is
optimal_k = 3


# Apply K-means clustering with the optimal K
from umap import UMAP
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler


# Drop rows with missing values in the selected features
features_for_clustering = ['Yearly_avg_view_on_travel_page', 'yearly_avg_Outstation_checkins',
'Daily_Avg_mins_spend_on_traveling_page']
df_clustering = df[features_for_clustering].dropna()


# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_clustering)


# Apply UMAP for dimensionality reduction (reduce to 2D for visualization)
umap = UMAP(n_components=2, random_state=42)
X_umap = umap.fit_transform(X_scaled)


# Apply K-Means clustering on the reduced data
optimal_k = 3
kmeans = KMeans(n_clusters=optimal_k, init='k-means++', random_state=42)
clusters = kmeans.fit_predict(X_umap)
# Add UMAP and cluster results to the DataFrame
df_clustering['UMAP_1'] = X_umap[:, 0]
df_clustering['UMAP_2'] = X_umap[:, 1]
df_clustering['Cluster'] = clusters
# Visualize the clusters
plt.figure(figsize=(12, 8))
sns.scatterplot(x='UMAP_1', y='UMAP_2', hue='Cluster', palette='viridis', data=df_clustering, s=100)
plt.title('Customer Segmentation using UMAP and K-Means Clustering')
plt.xlabel('UMAP Dimension 1'
plt.ylabel('UMAP Dimension 2') plt.legend(title='Cluster') plt.show()
```

```
# Merge the clustering results back to the original dataframe
df = pd.merge(df, df_clustering[['Cluster']], left_index=True, right_index=True, how='left')

# Visualize the clusters
sns.pairplot(df, hue='Cluster', palette='viridis', diag_kind='kde', vars=features_for_clustering)
plt.suptitle('Customer Segmentation using K-means Clustering', y=1.02)
plt.show()
```

3.3 Hyperparameter Tuning

Hyperparameter tuning is critical to optimize model performance. We employ:

- **Grid Search:** Exhaustively tests parameter combinations (e.g., K in K-Means, depth in Decision Trees).
- **Random Search:** Randomly selects parameter sets, allowing faster exploration.
- **Bayesian Optimization:** Uses probabilistic models to efficiently find optimal parameters.

Source code:

```
umap = UMAP(n_components=2,
n_neighbors=n_neighbors, random_state=42)
X_umap = umap.fit_transform(X_scaled)
kmeans = KMeans(n_clusters=k, init='k-means++',
random_state=42)
df_clustering['Cluster'] = kmeans.fit_predict(X_umap)
```

3.4 Model Evaluation Metrics

The performance of the models is evaluated using the following metrics:

1. **Clustering Evaluation:**
   - **Silhouette Score:** Measures how well-separated clusters are.
   - **Adjusted Rand Index (ARI):** Compares clustering results to ground truth labels.
2. **Classification Evaluation:**
   - **Accuracy:** Measures correct predictions.
   - **Precision & Recall:** Evaluate false positives and false negatives.
   - **F1-Score:** Balances precision and recall.

- **R-Squared (R^2):** Evaluates model fit in regression problems.
- **Mean Squared Error (MSE):** Measures prediction error.

3.5 Cross-Validation

Cross-validation is applied to ensure that the model generalizes well to unseen data.

- **K-Fold Cross-Validation:** Splits data into K subsets and trains the model on different combinations.
- **Stratified K-Fold (for imbalanced data):** Ensures proportional representation in each fold.

Source code:

```
# Determine the optimal number of clusters using the Elbow Method
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)

# Plot the Elbow Method graph
plt.figure(figsize=(14, 6))
plt.plot(range(1, 11), wcss, marker='o', linestyle='--')
plt.title('Elbow Method for Optimal K')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Within-Cluster Sum of Squares (WCSS)')
plt.show()
```

3.6 Conclusion of Phase

In Phase 3, we trained models using clustering and dimensionality reduction techniques to analyze customer journeys. Various algorithms, including PCA, Autoencoders, K-Means, Decision Trees, and Neural Networks, were applied to improve segmentation and personalization. Hyperparameter tuning methods like grid search, random search, and Bayesian optimization were used to optimize model performance. Evaluation metrics such as silhouette score, accuracy, precision, recall, F1score, MSE, and R^2 were applied to measure effectiveness. Cross-validation was performed to ensure model generalization.