Market Segmentation Analysis using ML

Project Title : Market Segmentation Using Machine Learning

1. **INTRODUCTION:**

Project Overview - In this report, we are going through the Steps of Market segmentation and covering the basic idea of market segmentation. We will be covering all the instruction or procedure you have to keep in mind during the market segmentation.

Purpose - The purpose of marketing is to match the genuine needs and desires of consumers with the offers of suppliers particularly suited to satisfy those needs and desires. This matching process benefits consumers and suppliers, and drives an organization's marketing planning process.

2. **LITERATURE SURVEY:**

Existing problem - The existing problem that this project likely aims to address could be the inefficiency or limitations of traditional segmentation methods like the inability to handle large datasets , manual segmentation , Difficulty in Identifying Subtle Patterns and many more.

References

- Supervised learning: https://www.javatpoint.com/supervised- machinelearningo

Unsupervised learning: https://www.javatpoint.com/unsupervised-machine-learning

Problem Statement Definition -The market segmentation project aimed to leverage machine learning techniques to analyze customer data and identify distinct market segments based on behavior, preferences, and characteristics. By applying unsupervised machine learning algorithms, the project sought to uncover hidden patterns within the data and provide actionable insights for targeted marketing and product development strategies.

3. **IDEATION & PROPOSED SOLUTION**

Empathy Map Canvas - uploaded in the git repo
Ideation & Brainstorming - uploaded in the git repo

4. **REQUIREMENT ANALYSIS:**

Functional requirement -
Non-Functional requirements -

5. **PROJECT DESIGN**:
   5.1 Data Flow Diagrams & User Stories - uploaded in git repo
   5.2 Solution Architecture - uploaded in the git repo

6. **PROJECT PLANNING & SCHEDULING:** uploaded in the git repo
   6.1 Technical Architecture
   6.2 Sprint Planning & Estimation
   6.3 Sprint Delivery Schedule

7. **CODING & SOLUTIONING - uploaded in the git repo**

8. PERFORMANCE TESTING

8.1 Performace Metrics - Hyper parameter tuning for the random forest classifier and for KNN
- cross validation scores for optimal parameters

Cross Validation

[89]:
```
from sklearn.model_selection import cross_val_score
cv_values = [3, 5, 7, 10, 12, 15]
cv_scores = {}
# Iterate over cv values and evaluate model performance
for cv in cv_values:
    knn = KNeighborsClassifier(n_neighbors=1)  # Set desired hyperparameters
    scores = cross_val_score(knn, x_scaled, y_train_smote, cv=cv)
    cv_scores[cv] = scores.mean()

# Print mean scores for each cv value
for cv, mean_score in cv_scores.items():
    print(f"CV: {cv}, Mean Accuracy: {mean_score}")

# Select the best cv value based on the scores
best_cv = max(cv_scores, key=cv_scores.get)
print(f"\nBest CV value: {best_cv} with mean accuracy: {cv_scores[best_cv]}")
```

```
CV: 3, Mean Accuracy: 0.5474833808167141
CV: 5, Mean Accuracy: 0.5551192714255159
CV: 7, Mean Accuracy: 0.5551131150134472
CV: 10, Mean Accuracy: 0.5547596479350034
CV: 12, Mean Accuracy: 0.5562391774891774
CV: 15, Mean Accuracy: 0.5567477203647417

Best CV value: 15 with mean accuracy: 0.5567477203647417
```

Hyper Tuning

```
# Define the parameter grid to search through
param_grid = {
    'n_neighbors': [1,2,3, 5, 7, 9,20,22,36,40,54,65,70],  # values of K to try
    'weights': ['uniform', 'distance'],  # weight options
    'metric': ['euclidean', 'manhattan','minoskwi']  # distance metrics
}
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
# Create a KNN classifier
knn = KNeighborsClassifier()
# Use GridSearchCV to search for the best parameters
grid_search = GridSearchCV(knn, param_grid, cv=15, scoring='accuracy')
grid_search.fit(x_train, y_train)
# Print the best parameters found
print("Best parameters:", grid_search.best_params_)
# Get the best model
best_knn = grid_search.best_estimator_
# Evaluate the model on the test set
test_accuracy = best_knn.score(x_test, y_test)
print("Test set accuracy:", test_accuracy)
```

```
Best parameters: {'metric': 'manhattan', 'n_neighbors': 5, 'weights': 'distance'}
Test set accuracy: 0.542654028436019
```

9. RESULTS

Testing of the model

```
random.predict([[-1.04667205,0.40519021 ,-0.24033293 , 0.44185143, -0.94725778,  0.39961435,
   0.98398389, -1.22401824, -0.64740827, -0.41702883, -0.53892595,  0.72370044,
  -1.47849323 ,-0.76618433]])
```

```
0]: array([3])
```

9.1 Output Screenshots

```
In [176]:
rdf=RandomForestClassifier(n_estimators = 300,random_state = 42,n_jobs=-1, min_samples_leaf = 1,min_samples_split= 5)
random=rdf.fit(x_train,y_train)
predict_random=random.predict(x_test)
predict_model=random.predict(x_train)
training_accuracy=random.score(x_train,y_train)
testing_accuracy=random.score(x_test,y_test)
print("*** Random Forest ***")
print("Training Accuracy : ",training_accuracy)
print("Testing Accuracy : ",testing_accuracy)
print("Accuracy Score : ",accuracy_score(y_test, predict_random))
print("** Confusion Matrix **")
print(confusion_matrix(y_test, predict_random))
print("** Classification Report **")
print(classification_report(y_test, predict_random))
```

```
*** Random Forest ***
Training Accuracy :  0.9263657957244655
```

```
*** Random Forest ***
Training Accuracy :  0.9263657957244655
Testing Accuracy :  0.5971563981042654
Accuracy Score :  0.5971563981042654
** Confusion Matrix **
[[32  2  5 17  3 12]
 [ 2 59  0  2  7  0]
 [ 4  0 60  1  0  5]
 [17  6  1 22 19  5]
 [ 3  7  1 16 38  6]
 [ 9  0 15  4  1 41]]
** Classification Report **
              precision    recall  f1-score   support

           0       0.48      0.45      0.46        71
           1       0.80      0.84      0.82        70
           2       0.73      0.86      0.79        70
           3       0.35      0.31      0.33        70
           4       0.56      0.54      0.55        71
           5       0.59      0.59      0.59        70

    accuracy                           0.60       422
   macro avg       0.59      0.60      0.59       422
weighted avg       0.59      0.60      0.59       422
```

XGBoost Classifier

0]:

```python
import xgboost as xgb
def xgboost(x_train,x_test,y_train,y_test):
    xgb_classifier = xgb.XGBClassifier(objective='multi:softmax', num_class=5)  # Assuming 5 classes for the target variable

# Training the classifier
    xgb_classifier.fit(x_train, y_train)

# Making predictions on the test set
    y_pred = xgb_classifier.predict(x_test)

# Evaluating the model
    accuracy = accuracy_score(y_test, y_pred)
    print("Accuracy:", accuracy)
    print("*** XGBoost  ***")
    print("Training Accuracy : ",training_accuracy)
    print("Testing Accuracy : ",testing_accuracy)
    print("Accuracy Score : ",accuracy_score(y_test, y_pred))
    print("** Confusion Matrix **")
    print(confusion_matrix(y_test,y_pred))
    print("** Classification Report **")
    print(classification_report(y_test,y_pred))
```

```
Accuracy: 0.5971563981042654
*** XGBoost  ***
Training Accuracy :  0.9263657957244655
Testing Accuracy :  0.5971563981042654
Accuracy Score :  0.5971563981042654
** Confusion Matrix **
[[34  1  3 16  3 14]
 [ 3 57  0  2  8  0]
 [ 5  0 57  1  0  7]
 [21  3  3 21 19  3]
 [ 5  7  1 15 42  1]
 [12  0 10  4  3 41]]
** Classification Report **
              precision    recall  f1-score   support

           0       0.42      0.48      0.45        71
           1       0.84      0.81      0.83        70
           2       0.77      0.81      0.79        70
           3       0.36      0.30      0.33        70
```

```
** Classification Report **
              precision    recall  f1-score   support

           0       0.42      0.48      0.45        71
           1       0.84      0.81      0.83        70
           2       0.77      0.81      0.79        70
           3       0.36      0.30      0.33        70
           4       0.56      0.59      0.58        71
           5       0.62      0.59      0.60        70

    accuracy                           0.60       422
   macro avg       0.60      0.60      0.60       422
weighted avg       0.59      0.60      0.59       422

----------------------------------------------------------------------
```

10. ADVANTAGES & DISADVANTAGES

The advantages are as below :

i. Targeted Marketing - By identifying distinct customer segments, businesses can tailor their marketing strategies to specific groups, leading to more effective and personalized campaigns.

ii. Product Customization- Understanding different market segments allows businesses to customize their products or services to better meet the specific needs and preferences of each segment, leading to higher customer satisfaction.

iii. Improved Customer Retention - By catering to the unique preferences of different customer segments, businesses can enhance customer loyalty and retention, leading to long-term profitability.

iv. Optimized Resource Allocation - Market segmentation helps businesses allocate their resources more efficiently by focusing on the most profitable customer segments, leading to improved ROI.

v. Enhanced Competitive Advantage - By understanding the market at a granular level, businesses can gain a competitive edge by offering unique value propositions tailored to specific customer segments.

Overall, market segmentation projects enable businesses to better understand their customers, improve their marketing efforts, and drive sustainable growth by catering to the diverse needs of different customer segments.

Now discussing about the disadvantages, they are mainly about:

i. Data Privacy Concerns - The collection and analysis of customer data for segmentation

purposes may raise privacy concerns and regulatory compliance issue

ii. Complexity - Implementing a market segmentation project can be complex and resource-intensive, requiring advanced data analysis and segmentation techniques

iii. Overlooking Overlapping Segments - There is a risk of overlooking overlapping segments or failing to account for customers who exhibit behaviors that span multiple segments

iv. Customer Alienation - Overly targeted marketing efforts based on segmentation can sometimes alienate customers

11. CONCLUSION - Here with the code we have addressed and solved many problems and made it easier for the companies to understand the patterns in the customer visit frequency which would increase the sales of the company. We have used different classification algorithms to get the best accurate results from the datatset provided.

12. FUTURE SCOPE

Real-Time Adaptability

Integration of real-time data streams and advanced algorithms to enable dynamic segmentation, capturing immediate shifts in consumer behavior and preferences for agile decision-making.

Personalized Customer Experiences

Utilization of segmented insights to craft highly personalized customer experiences across multiple touchpoints, fostering deeper connections and increased brand loyalty.

Predictive Analytics

Expansion of the project's capabilities towards predictive modeling, forecasting future trends, and anticipating customer needs, enabling proactive marketing and product development strategies.

Integration with Emerging Technologies

Exploration and integration of emerging technologies such as AI-driven chatbots, augmented reality, or IoT devices to further personalize interactions based on segmented insights.

13. APPENDIX

GitHub & Project Demo Link - https://github.com/harshitha-mandula/Market-segmentation

Project Demo link - https://drive.google.com/file/d/1K4wPjsALwzcRmciHtWt-LgI-rr01YcBR/view?usp=sharing