Taxonomy Development for Customer Support Intents

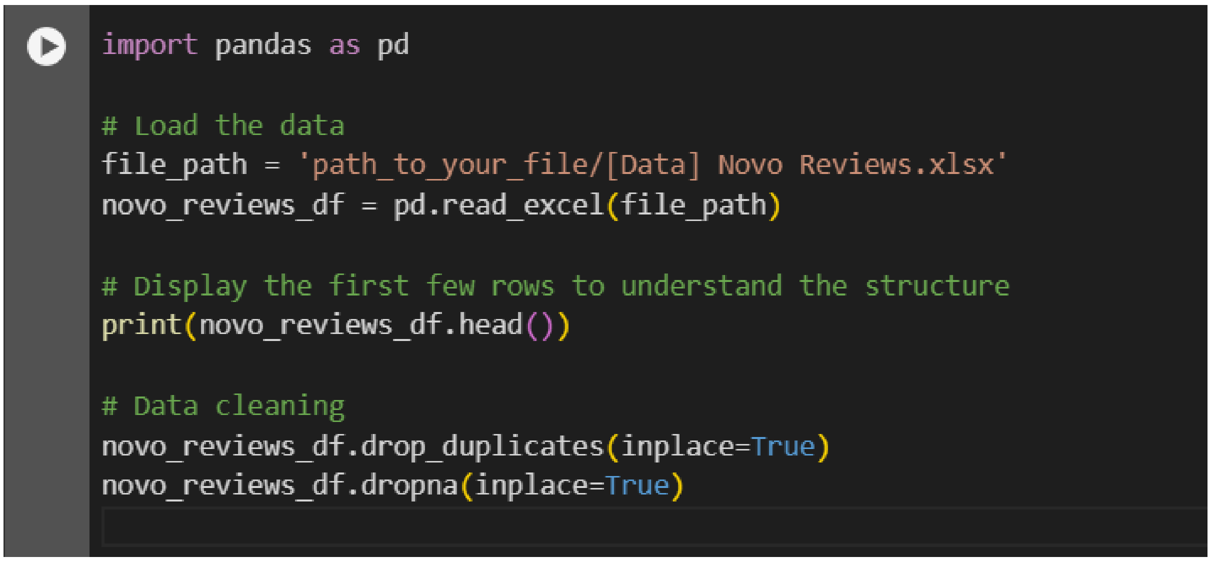
Objective:
The primary objective of this stage is to develop a comprehensive and hierarchical taxonomy of customer support intents based on the provided customer feedback data. This taxonomy will serve as the foundation for classifying customer feedback into specific categories, enabling more efficient and accurate issue resolution.

Approach:

1. Data Understanding:

First, load and examine the customer feedback data to identify common themes and issues. This involves:

- Loading the dataset and inspecting its structure.
- Performing initial data cleaning, such as removing duplicates and handling missing values



```python
import pandas as pd

# Load the data
file_path = 'path_to_your_file/[Data] Novo Reviews.xlsx'
novo_reviews_df = pd.read_excel(file_path)

# Display the first few rows to understand the structure
print(novo_reviews_df.head())

# Data cleaning
novo_reviews_df.drop_duplicates(inplace=True)
novo_reviews_df.dropna(inplace=True)
```

2. Research Existing Taxonomies

Study existing customer support taxonomies in similar domains (e.g., banking, marketplaces) to identify standard categories and subcategories. Utilize the examples provided in the assignment as a reference.

Examples of Taxonomies in Similar Domains:

- Banking:
    - Account Issues: Login Problems, Password Recovery
    - Transaction Issues: Failed Transactions, Unauthorized Transactions
    - Card Issues: Card Declined, Lost/Stolen Card

- Marketplaces:
    - Orders: Order Placement, Order Status
    - Payments: Payment Failed, Refund Issues
    - Delivery: Late Delivery, Wrong Item Delivered

3. Create Initial Categories

Based on the data examination and research, create a preliminary set of categories and subcategories. The categories should be broad enough to cover all types of feedback but specific enough to allow precise classification.

Proposed Taxonomy:

- Account Issues
    - Login Problems
    - Password Recovery

- Transaction Issues
    - Failed Transactions
    - Unauthorized Transactions

- Card Issues
    - Card Declined
    - Lost/Stolen Card

- Service Requests
    - Feature Requests
    - Account Changes

- Invoice Issues
    - Sent but Unpaid
    - Sent and Paid
    - Conflict/Dispute

4. Iterate and Refine

Refine the taxonomy based on feedback and further analysis to ensure it comprehensively covers all types of customer feedback. This iterative process may involve:

- Reviewing additional customer feedback to identify any missing categories.
- Seeking input from domain experts to validate the taxonomy.
- Adjusting categories and subcategories to improve clarity and coverage.

Example Refinement Process:

Review Additional Feedback:

- Collect and analyze more feedback data to identify any new issues.

Seek Expert Input:

- Discuss the taxonomy with customer support experts to ensure its relevance and completeness.

Adjust Categories:

- Modify or add categories based on new insights and expert recommendations.

Justifications for Category Choices

1.Account Issues:

- Login Problems: Common issue where users face difficulties logging into their accounts.
- Password Recovery: Frequent request related to recovering forgotten or lost passwords.

2. Transaction Issues:

- Failed Transactions: Critical category for issues where transactions do not go through.
- Unauthorized Transactions: Important to address potential fraud and security concerns.

3. Card Issues:

- Card Declined: Common problem where cards are not accepted during transactions.
- Lost/Stolen Card: Security-related issues that need prompt attention.

4. Service Requests:

- Feature Requests: Feedback on desired features can guide product development.
- Account Changes: Requests related to updating or modifying account information.

5. Invoice Issues:

- Sent but Unpaid: Common issue where invoices are sent but remain unpaid.
- Sent and Paid: Feedback on completed transactions, possibly including complaints or confirmations.
- Conflict/Dispute: Disputes or conflicts related to invoices that need resolution.

Auto Extract Intents from Model (Optional/Stretch)

Objective: Explore methods to automatically extract intents from the customer feedback data using natural language processing (NLP) techniques. This step aims to identify underlying themes and patterns in the feedback without extensive manual intervention.

Approach:

1. Data Preprocessing
Before extracting intents, preprocess the text data to clean and normalize it. This involves:
Text Cleaning:

- Removing special characters, numbers, and punctuation.
- Converting text to lowercase to ensure uniformity.

Tokenization:

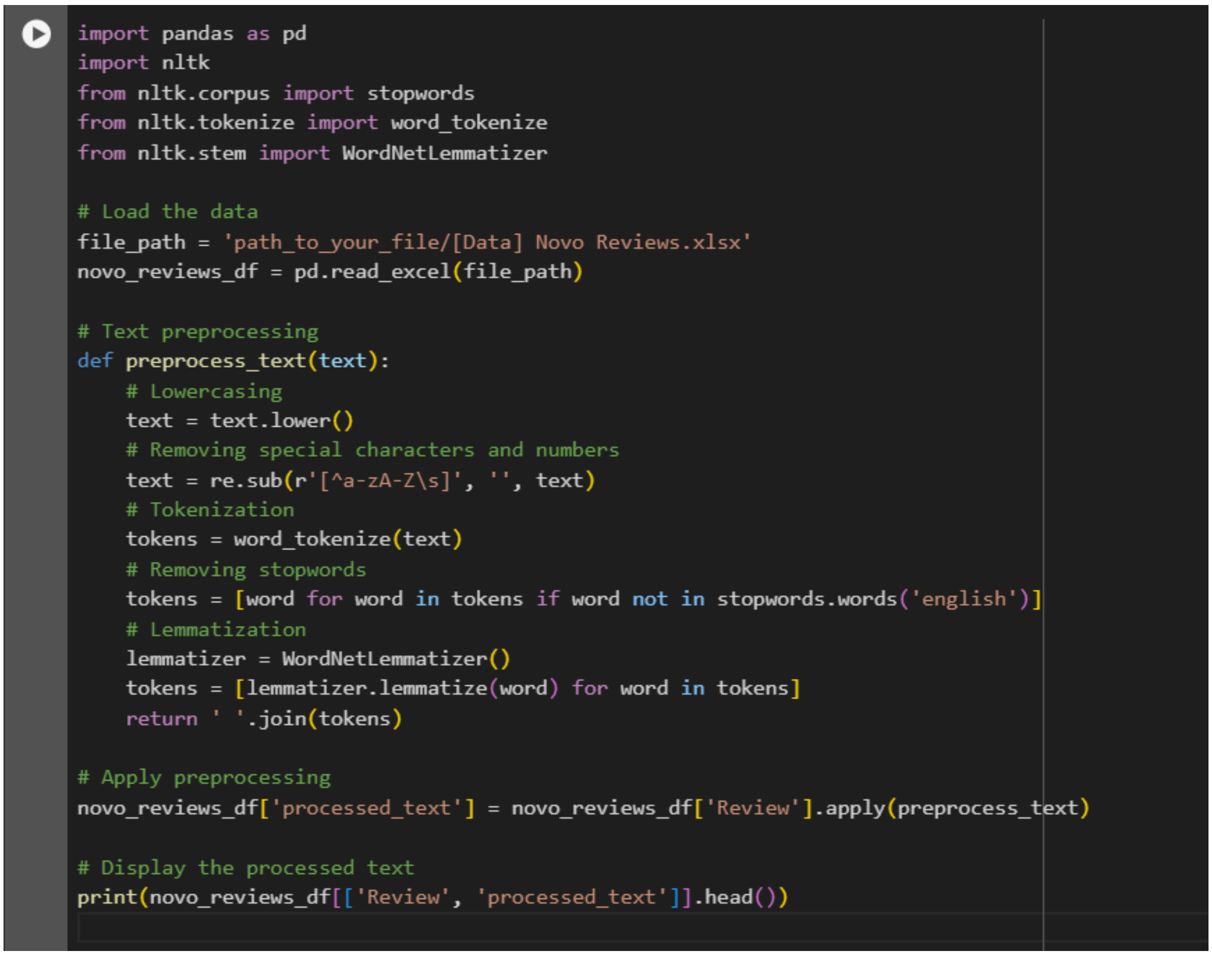
- Splitting the text into individual words or tokens.

Stop Words Removal:

- Removing common words (e.g., 'and', 'the', 'is') that do not contribute much to the meaning.

Lemmatization/Stemming:

- Reducing words to their base or root form.



```python
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

# Load the data
file_path = 'path_to_your_file/[Data] Novo Reviews.xlsx'
novo_reviews_df = pd.read_excel(file_path)

# Text preprocessing
def preprocess_text(text):
    # Lowercasing
    text = text.lower()
    # Removing special characters and numbers
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    # Tokenization
    tokens = word_tokenize(text)
    # Removing stopwords
    tokens = [word for word in tokens if word not in stopwords.words('english')]
    # Lemmatization
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]
    return ' '.join(tokens)

# Apply preprocessing
novo_reviews_df['processed_text'] = novo_reviews_df['Review'].apply(preprocess_text)

# Display the processed text
print(novo_reviews_df[['Review', 'processed_text']].head())
```

2. Topic Modeling:

Use topic modeling techniques like Latent Dirichlet Allocation (LDA) to discover hidden topics within the feedback data. LDA helps in identifying groups of words that frequently occur together, which can represent different intents.

Steps:

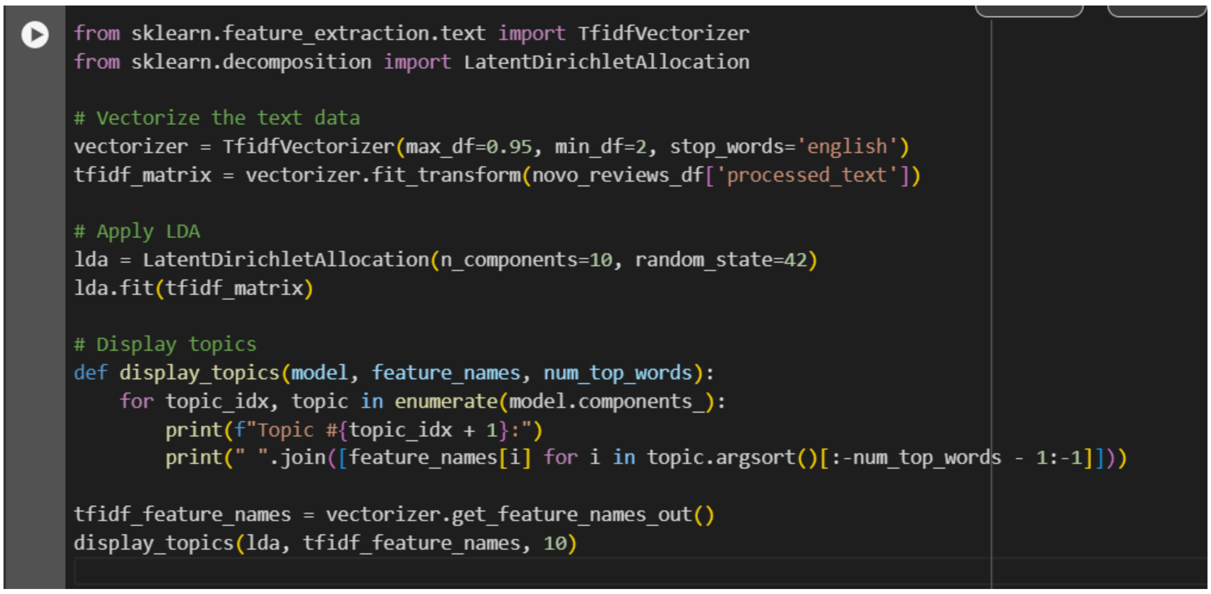
Vectorize the Text:

- Convert the text data into numerical format using methods like TF-IDF (Term Frequency-Inverse Document Frequency).

Apply LDA:

- Use LDA to extract topics from the vectorized text data.

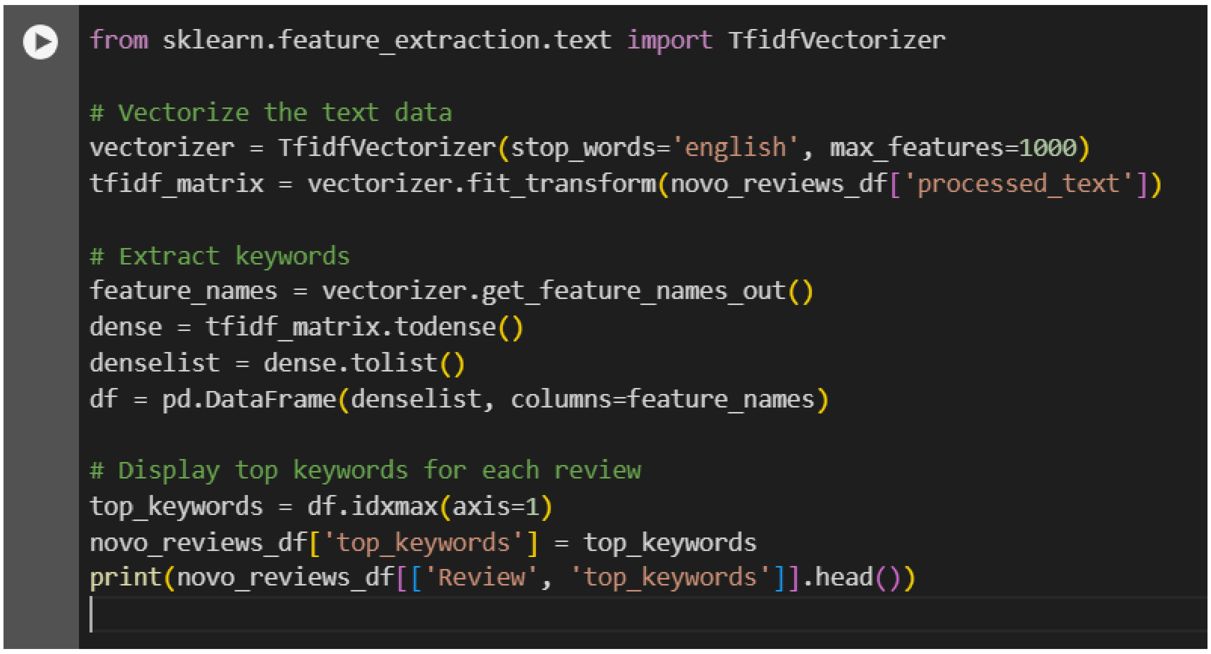
Example Code:



```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import LatentDirichletAllocation

# Vectorize the text data
vectorizer = TfidfVectorizer(max_df=0.95, min_df=2, stop_words='english')
tfidf_matrix = vectorizer.fit_transform(novo_reviews_df['processed_text'])

# Apply LDA
lda = LatentDirichletAllocation(n_components=10, random_state=42)
lda.fit(tfidf_matrix)

# Display topics
def display_topics(model, feature_names, num_top_words):
    for topic_idx, topic in enumerate(model.components_):
        print(f"Topic #{topic_idx + 1}:")
        print(" ".join([feature_names[i] for i in topic.argsort()[:-num_top_words - 1:-1]]))

tfidf_feature_names = vectorizer.get_feature_names_out()
display_topics(lda, tfidf_feature_names, 10)
```

3. Keyword Extraction

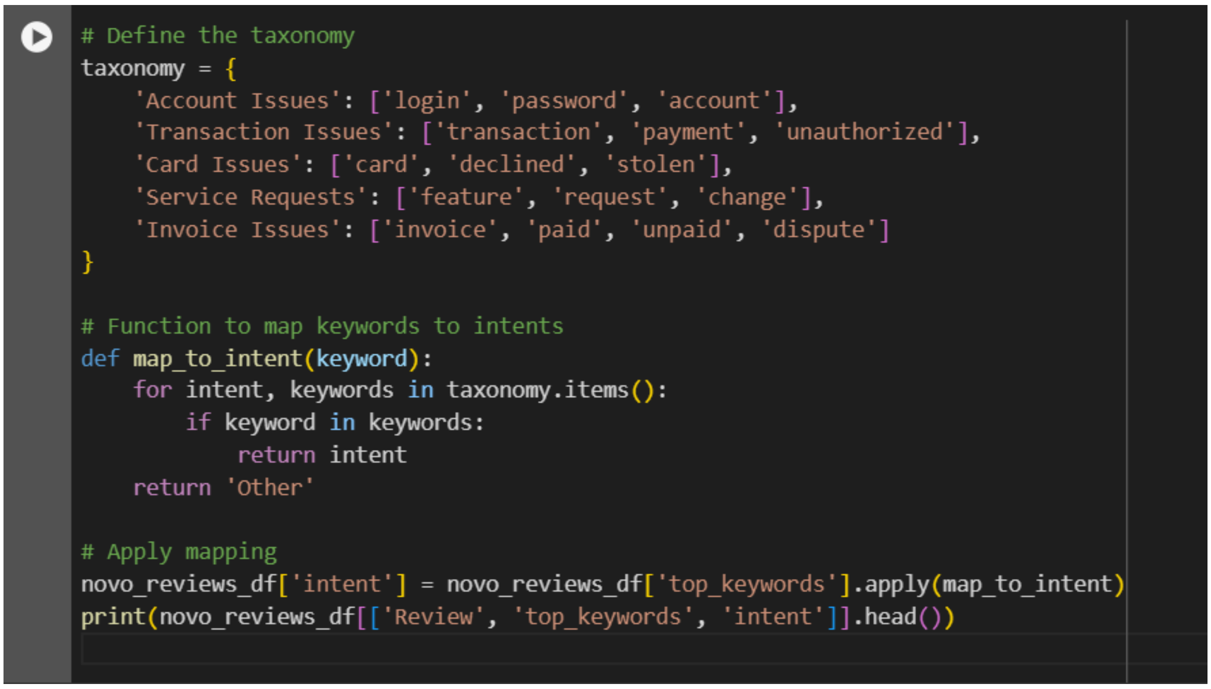
Extract significant keywords from the feedback data to understand the primary themes. Techniques like TF-IDF and RAKE (Rapid Automatic Keyword Extraction) can be used.



```python
from sklearn.feature_extraction.text import TfidfVectorizer

# Vectorize the text data
vectorizer = TfidfVectorizer(stop_words='english', max_features=1000)
tfidf_matrix = vectorizer.fit_transform(novo_reviews_df['processed_text'])

# Extract keywords
feature_names = vectorizer.get_feature_names_out()
dense = tfidf_matrix.todense()
denselist = dense.tolist()
df = pd.DataFrame(denselist, columns=feature_names)

# Display top keywords for each review
top_keywords = df.idxmax(axis=1)
novo_reviews_df['top_keywords'] = top_keywords
print(novo_reviews_df[['Review', 'top_keywords']].head())
```

4. Mapping Keywords to Intents

Map the extracted keywords and topics to the predefined taxonomy of intents. This can be done by creating a mapping dictionary or using similarity measures.



```python
# Define the taxonomy
taxonomy = {
    'Account Issues': ['login', 'password', 'account'],
    'Transaction Issues': ['transaction', 'payment', 'unauthorized'],
    'Card Issues': ['card', 'declined', 'stolen'],
    'Service Requests': ['feature', 'request', 'change'],
    'Invoice Issues': ['invoice', 'paid', 'unpaid', 'dispute']
}

# Function to map keywords to intents
def map_to_intent(keyword):
    for intent, keywords in taxonomy.items():
        if keyword in keywords:
            return intent
    return 'Other'

# Apply mapping
novo_reviews_df['intent'] = novo_reviews_df['top_keywords'].apply(map_to_intent)
print(novo_reviews_df[['Review', 'top_keywords', 'intent']].head())
```

Output:

- Summary of methods used for automatic intent extraction and their effectiveness.
- List of identified topics and keywords mapped to the predefined taxonomy.

Teaching the Model to Classify Tickets According to Defined Intents

Approach:

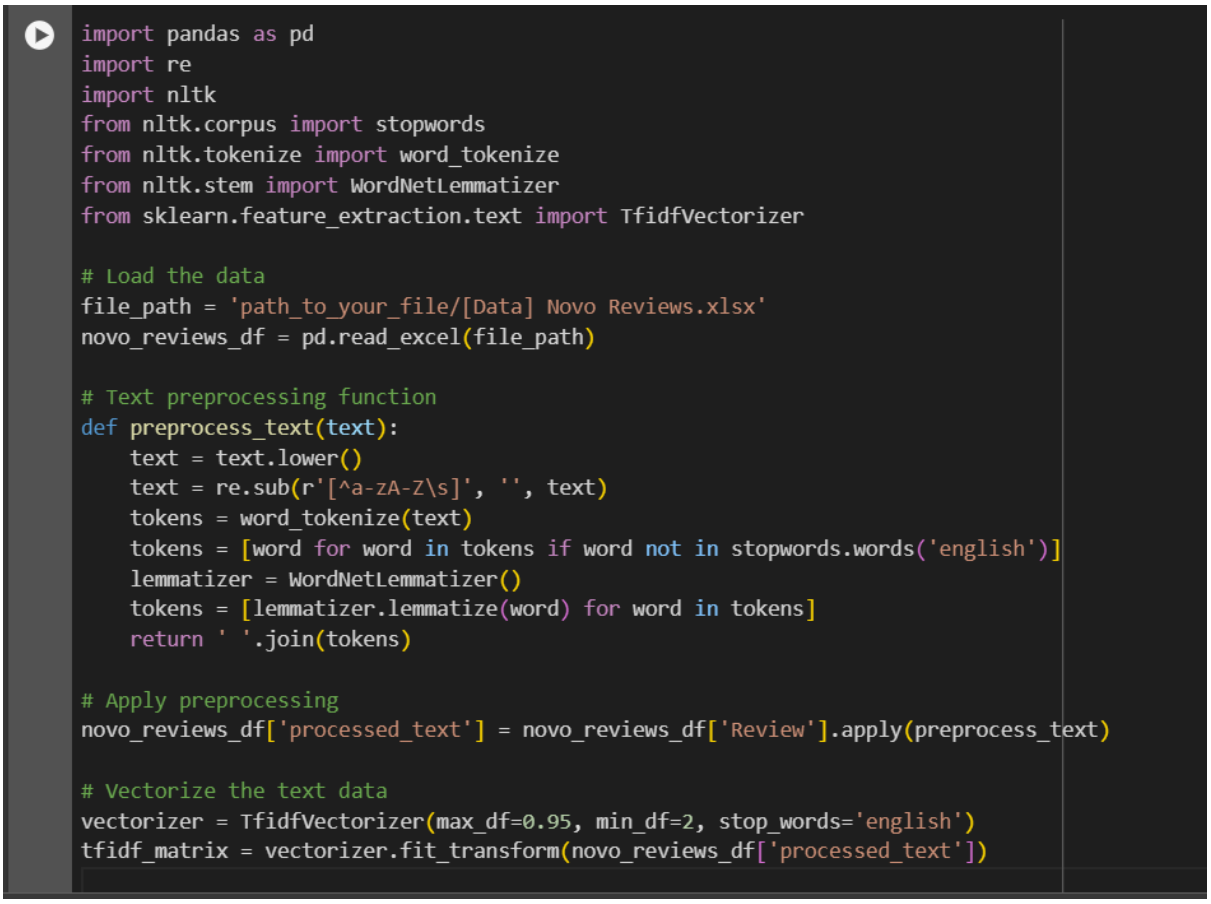
1. Data Preprocessing

Before training the model, preprocess the data to ensure it is clean and suitable for training. This includes text cleaning, tokenization, and vectorization.

Steps:

- Text Cleaning: Remove special characters, numbers, and punctuation. Convert text to lowercase.
- Tokenization: Split the text into individual words or tokens.
- Stop Words Removal: Remove common words that do not contribute much to the meaning.
- Lemmatization/Stemming: Reduce words to their base or root form.

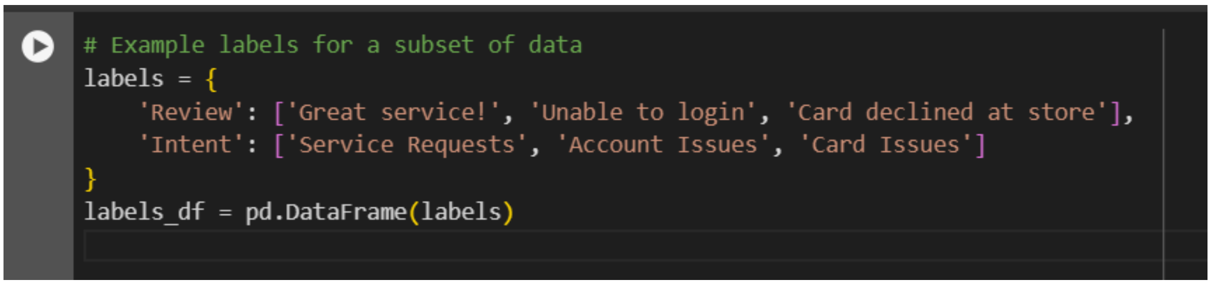
- Vectorization: Convert the text data into numerical format using TF-IDF or word embeddings.



```python
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer

# Load the data
file_path = 'path_to_your_file/[Data] Novo Reviews.xlsx'
novo_reviews_df = pd.read_excel(file_path)

# Text preprocessing function
def preprocess_text(text):
    text = text.lower()
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    tokens = word_tokenize(text)
    tokens = [word for word in tokens if word not in stopwords.words('english')]
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]
    return ' '.join(tokens)

# Apply preprocessing
novo_reviews_df['processed_text'] = novo_reviews_df['Review'].apply(preprocess_text)

# Vectorize the text data
vectorizer = TfidfVectorizer(max_df=0.95, min_df=2, stop_words='english')
tfidf_matrix = vectorizer.fit_transform(novo_reviews_df['processed_text'])
```

2. Labeling Data

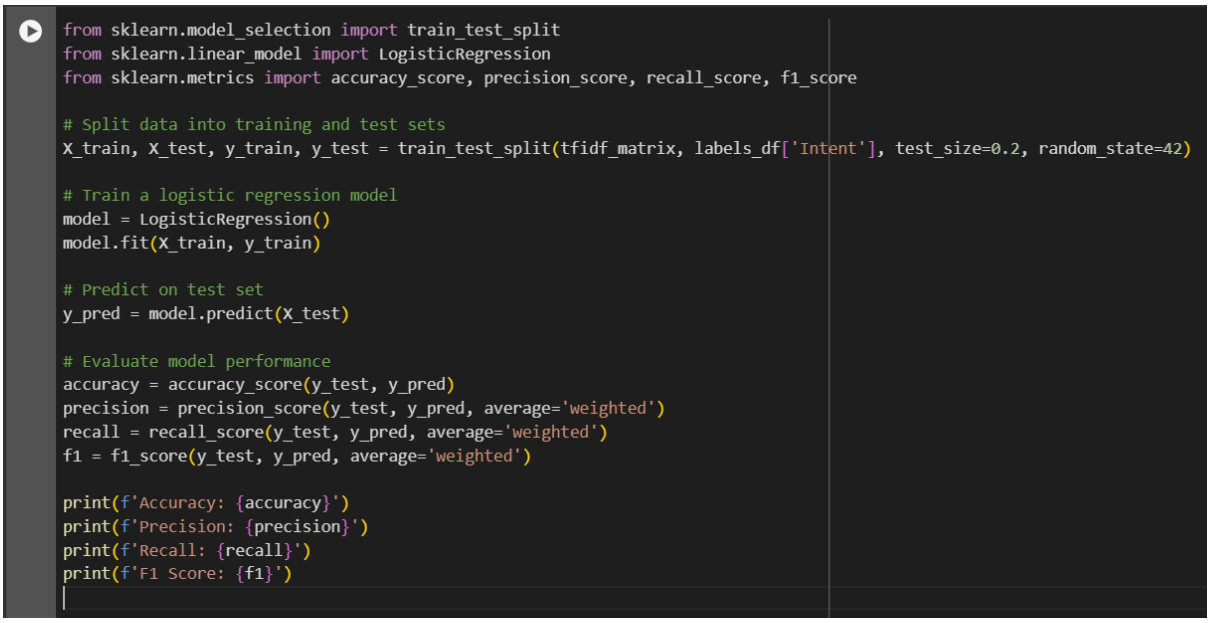
Manually label a representative sample of feedback data based on the taxonomy. This labeled data will be used to train the model.



```python
# Example labels for a subset of data
labels = {
    'Review': ['Great service!', 'Unable to login', 'Card declined at store'],
    'Intent': ['Service Requests', 'Account Issues', 'Card Issues']
}
labels_df = pd.DataFrame(labels)
```

3. Model Selection

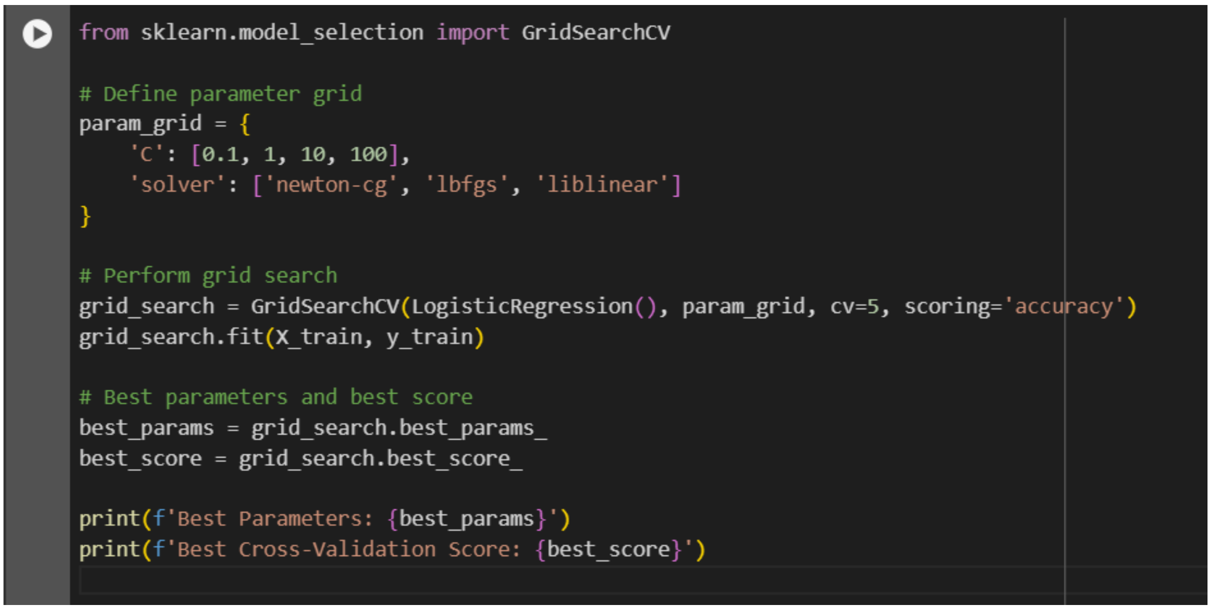
Experiment with different models to identify the best one for classifying customer feedback into the predefined intents. Consider models like Logistic Regression, Support Vector Machines (SVM), Random Forest, and advanced models like BERT.



```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(tfidf_matrix, labels_df['Intent'], test_size=0.2, random_state=42)

# Train a logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Predict on test set
y_pred = model.predict(X_test)

# Evaluate model performance
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')
```

4. Hyperparameter Tuning

Optimize model performance through hyperparameter tuning using techniques like grid search or random search.



```python
from sklearn.model_selection import GridSearchCV

# Define parameter grid
param_grid = {
    'C': [0.1, 1, 10, 100],
    'solver': ['newton-cg', 'lbfgs', 'liblinear']
}

# Perform grid search
grid_search = GridSearchCV(LogisticRegression(), param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)

# Best parameters and best score
best_params = grid_search.best_params_
best_score = grid_search.best_score_

print(f'Best Parameters: {best_params}')
print(f'Best Cross-Validation Score: {best_score}')
```

5. Model Evaluation

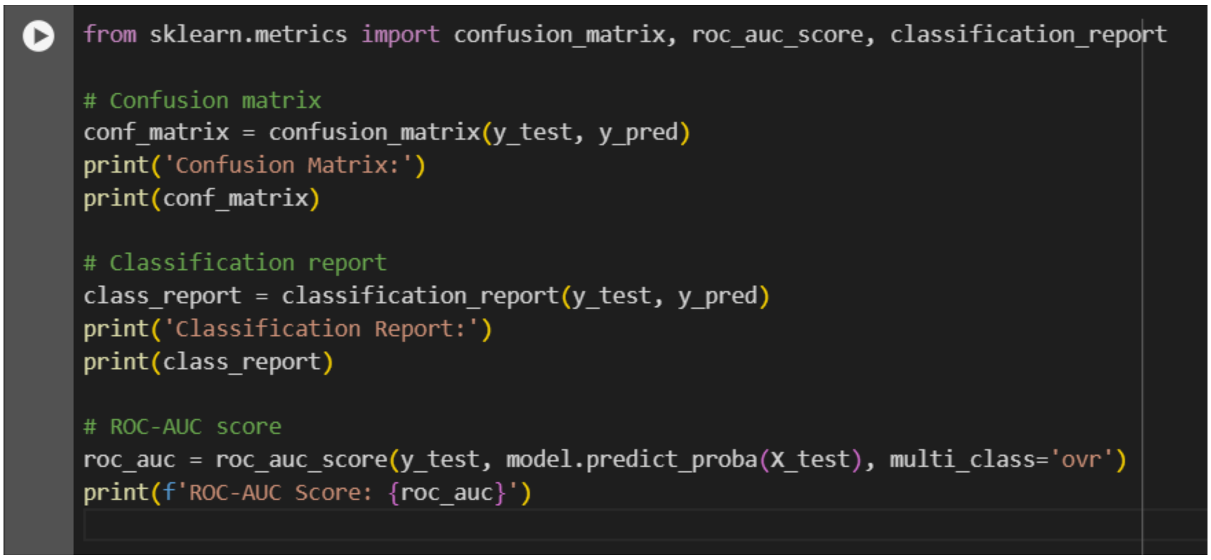
Define evaluation criteria and methods for validating the model's performance.

Evaluation Metrics:

- Accuracy: Overall correctness of the model.
- Precision: Proportion of true positive predictions.
- Recall: Proportion of actual positives correctly identified.
- F1-Score: Harmonic mean of precision and recall.

Validation Techniques:

- Cross-Validation: Use k-fold cross-validation to ensure robustness.
- Confusion Matrix: Analyze the confusion matrix to understand performance across different classes.
- ROC-AUC: Evaluate the model's ability to distinguish between classes.



```python
from sklearn.metrics import confusion_matrix, roc_auc_score, classification_report

# Confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print('Confusion Matrix:')
print(conf_matrix)

# Classification report
class_report = classification_report(y_test, y_pred)
print('Classification Report:')
print(class_report)

# ROC-AUC score
roc_auc = roc_auc_score(y_test, model.predict_proba(X_test), multi_class='ovr')
print(f'ROC-AUC Score: {roc_auc}')
```

Output

- Detailed description of the model development process, including data preprocessing, model selection, training, hyperparameter tuning, and evaluation results.
- Performance metrics and validation results demonstrating the effectiveness of the model.