```
import pandas as pd
import numpy as np
import re
from nltk.stem import WordNetLemmatizer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.neural_network import MLPClassifier
from lime.lime_text import LimeTextExplainer
import matplotlib.pyplot as plt

# Load and Preprocess the dataset
file_path = '/kaggle/input/requirment-csv/requirment.csv'
df = pd.read_csv(file_path, encoding='latin1')
lemmatizer = WordNetLemmatizer()

def clean_text(text):
    text = text.lower()
    text = re.sub(r'[^\w\s]', '', text)
    text = re.sub(r'\d+', '', text)
    return ' '.join([lemmatizer.lemmatize(word) for word in text.split()])

df['cleaned_reviews'] = df['Base_Reviews'].apply(clean_text)

# Prepare the data
vectorizer = CountVectorizer(max_features=1000)
X = vectorizer.fit_transform(df['cleaned_reviews'])
y_dict = {'feature': 0, 'issue': 1, 'user_experience': 2,
'other_information': 3}
df['category_id'] = df['category'].map(y_dict)
y = df['category_id'].values

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Define and train the MLP model
mlp_model = MLPClassifier(hidden_layer_sizes=(100,), max_iter=300,
random_state=42)
mlp_model.fit(X_train, y_train)

# Initialize LIME Text Explainer
explainer = LimeTextExplainer(class_names=list(y_dict.keys()))

# Function to predict with the MLP model
def mlp_predict_proba(texts):
    vec_texts = vectorizer.transform(texts)
    return mlp_model.predict_proba(vec_texts)

# Generate and plot LIME explanations for each class
for class_name in y_dict.keys():
    class_id = y_dict[class_name]
    indices = np.where(y_test == class_id)[0]
    if len(indices) > 0:
        idx = indices[0]  # First index of the specified class
        text_instance = df['cleaned_reviews'].iloc[idx]
        exp = explainer.explain_instance(text_instance, mlp_predict_proba,
num_features=10, labels=[class_id])

        # Extracting and plotting the explanation with values
        exp_list = exp.as_list(label=class_id)
        vals = [x[1] for x in exp_list]
        names = [x[0] for x in exp_list]
        vals.reverse()
        names.reverse()
        colors = ['orange' if x > 0 else 'blue' for x in vals]
        pos = np.arange(len(exp_list)) + .5

        # Create a combined figure with a table on the left and a bar plot
on the right
        fig, (ax_table, ax) = plt.subplots(nrows=1, ncols=2, figsize=(20,
8), gridspec_kw={'width_ratios': [1.2, 2]})

        # Add table to the left axis
        table_data = [[names[i],
vectorizer.transform([text_instance]).toarray()[0][vectorizer.vocabulary_[
names[i]]], vals[i]] for i in range(len(names))]
        table = ax_table.table(cellText=table_data, colLabels=["Feature",
"Value", "Weight"], cellLoc='center', loc='center')
        table.auto_set_font_size(False)
        table.set_fontsize(13)
        table.scale(1.5, 1.5)
        ax_table.axis('off')

        # Add bar plot to the right axis
        bars = ax.barh(pos, vals, align='center', color=colors)
        ax.set_yticks(pos)
        ax.set_yticklabels(names)
        ax.set_xlabel('Weight')
        ax.set_title(f'LIME Explanation for Class "{class_name}"')

        # Adding value annotations
        for bar, value in zip(bars, vals):
            ax.text(bar.get_width(), bar.get_y() + bar.get_height()/2,
f'{value:.2f}', va='center', ha='right' if value < 0 else 'left')

        plt.tight_layout(pad=3.0)
        plt.subplots_adjust(wspace=0.5)
        plt.savefig(f'lime14_explanation_{class_name}.png', dpi=300)
        plt.close()

print("Enhanced LIME explanations generated for each class.")
```