Python Code Explanation with Code Snippets

Imports

The necessary libraries are imported for various tasks:

- numpy, pandas: For numerical computations and data handling.
- scikit-learn: For preprocessing, training, and evaluation of ML models.
- xgboost: For using XGBoost classifier.
- nltk: For natural language processing.
- matplotlib, seaborn: For visualization.
- pickle: For saving and loading the trained model.

```
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from imblearn.over_sampling import SMOTE
import numpy as np
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
```

Data Loading and Preprocessing

The dataset is loaded, and missing values are replaced with empty strings. A 'content' column is created by combining 'author', 'title', and 'text' columns. The preprocess_text function cleans and tokenizes the text data.

```
news_dataset = pd.read_csv('data/train.csv')
news_dataset.fillna('', inplace=True)
news_dataset['content'] = news_dataset['author'] + ' ' + news_dataset['title'] + ' ' +
```

Python Code Explanation with Code Snippets

```
news_dataset['text']

lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))

def preprocess_text(content):
    content = re.sub('[^a-zA-Z]', ' ', content).lower()
    tokens = word_tokenize(content)
       lemmatized = [lemmatizer.lemmatize(word) for word in tokens if word not in
stop_words]
    return ' '.join(lemmatized)

news_dataset['content'] = news_dataset['content'].apply(preprocess_text)
```

Feature Extraction with TF-IDF

The text data is transformed into numerical features using TfidfVectorizer. It computes the Term Frequency-Inverse Document Frequency (TF-IDF) scores. The max_features parameter limits the vocabulary size to 5000, and ngram_range includes unigrams and bigrams.

```
vectorizer = TfidfVectorizer(max_features=5000, ngram_range=(1, 2))
X = vectorizer.fit_transform(news_dataset['content'])
```

Handling Class Imbalance with SMOTE

SMOTE is applied to oversample the minority class and balance the dataset. This helps improve the model's ability to classify both classes effectively.

```
smote = SMOTE(random_state=42)
X, y = smote.fit_resample(X, news_dataset['label'].values)
```

Model Training and Hyperparameter Tuning

Various models are trained and evaluated:

- Logistic Regression: Uses GridSearchCV to find optimal hyperparameters.

Python Code Explanation with Code Snippets

- Random Forest, XGBoost, and SVM are trained using default parameters.

Each model's accuracy is printed, and the best-performing model is saved.

```
log_reg = LogisticRegression()
grid_search = GridSearchCV(estimator=log_reg, param_grid={'C': [0.01, 0.1, 1]},
scoring='accuracy', cv=5)
grid_search.fit(X_train, y_train)
best_log_model = grid_search.best_estimator_

rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)

xgb_model = XGBClassifier(eval_metric='logloss', random_state=42)
xgb_model.fit(X_train, y_train)

svm_model = SVC(kernel='linear', random_state=42)
svm_model.fit(X_train, y_train)
```

Model Evaluation and Visualization

Accuracy, classification reports, and confusion matrices are generated for each model. Confusion matrices are visualized using heatmaps.

```
y_test_pred_rf = rf_model.predict(X_test)
print(classification_report(y_test, y_test_pred_rf))

def plot_confusion_matrix(y_true, y_pred, title):
    cm = confusion_matrix(y_true, y_pred)
      sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Fake', 'Real'],
yticklabels=['Fake', 'Real'])
    plt.title(title)
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()

plot_confusion_matrix(y_test, y_test_pred_rf, "Random Forest Confusion Matrix")
```

Python Code Explanation with Code Snippets

Saving the Best Model

The trained XGBoost model is saved to 'best_model.pkl' using the pickle module. This allows for reloading the model without retraining.

```
with open('best_model.pkl', 'wb') as file:
    pickle.dump(xgb_model, file)
print("Best model saved successfully!")
```