```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, f1_score, mean_squared_error
```

```
# Load the datasets
true_news_df = pd.read_csv("D:/Nishanth/True.csv")
fake_news_df = pd.read_csv("D:/Nishanth/Fake 2.csv")
```

```
# Concatenate datasets and create labels
true_news_df['label'] = 1
fake_news_df['label'] = 0
all_news = pd.concat([true_news_df, fake_news_df], ignore_index=True)
```

```
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(all_news['text'],
all_news['label'], test_size=0.2, random_state=42)
```

```
# Convert text data into TF-IDF vectors
tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

```
# Naive Bayes model
nb_model = MultinomialNB()
nb_model.fit(X_train_tfidf, y_train)
nb_pred = nb_model.predict(X_test_tfidf)
```

```
# Logistic Regression model
lr_model = LogisticRegression(max_iter=1000)
lr_model.fit(X_train_tfidf, y_train)
lr_pred = lr_model.predict(X_test_tfidf)
```

```
# SVM model
svm_model = SVC(kernel='linear')
svm_model.fit(X_train_tfidf, y_train)
svm_pred = svm_model.predict(X_test_tfidf)
```

```
svm_pred
```

```
# Evaluate models
def evaluate_model(y_true, y_pred, model_name):
    acc = accuracy_score(y_true, y_pred)
    f1 = f1_score(y_true, y_pred)
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    print(f"Evaluation metrics for {model_name}:")
    print(f"Accuracy: {acc:.4f}")
    print(f"F1 Score: {f1:.4f}")
    print(f"Mean Squared Error: {mse:.4f}")
    print(f"Root Mean Squared Error: {rmse:.4f}\n")
```

```
print("Evaluation Results:")
evaluate_model(y_test, nb_pred, "Naive Bayes")
evaluate_model(y_test, lr_pred, "Logistic Regression")
evaluate_model(y_test, svm_pred, "SVM")
```