Machine Learning Lab
(ISL65)

ASSIGNMENT

Fake News Detection

Submitted by
Student name – Sanjay Kumar SV
USN – 1MS21IS090
Student name – Suryansh Shivaprasad M
USN – 1MS21IS109

Under the guidance of
Prof. Jagadeesh Sai D
Assistant Professor
Department of ISE, MSRIT

RAMAIAH INSTITUTE OF TECHNOLOGY
(Autonomous Institute, affiliated to VTU) Accredited by the National Board of Accreditation &
NAAC with 'A+' Grade MSR Nagar, MSRIT Post, Bangalore-560054

ABSTRACT

In this project, we aim to build a robust machine-learning model for detecting fake news using a dataset sourced from Kaggle. The proliferation of misinformation in online platforms necessitates automated tools to distinguish between legitimate and fabricated news articles. We will preprocess the textual data, extract relevant features using TF-IDF and word embeddings, and experiment with various classification algorithms including Logistic Regression, Naive Bayes, SVM, Random Forest, and XGBoost. The performance of these models will be evaluated using standard metrics like accuracy, precision, recall, and F1-score.

Our approach involves comprehensive data preprocessing steps to clean and transform the raw text into numerical representations suitable for machine learning algorithms. We will explore different feature extraction techniques to capture the semantic meaning of news articles, enabling the models to discern patterns associated with fake news effectively. We aim to identify the most accurate and efficient model for this binary classification task by leveraging a diverse set of classifiers.

The final stage of our project will involve model deployment, where the best-performing classifier will be integrated into a functional system capable of real-time fake news detection. This deployment holds promise for aiding in the fight against misinformation by empowering users and platforms with a tool that can automatically flag potentially deceptive news articles. The project contributes to advancing the field of natural language processing and machine learning in the context of media integrity and information verification.

CONTENTS

1. Introduction

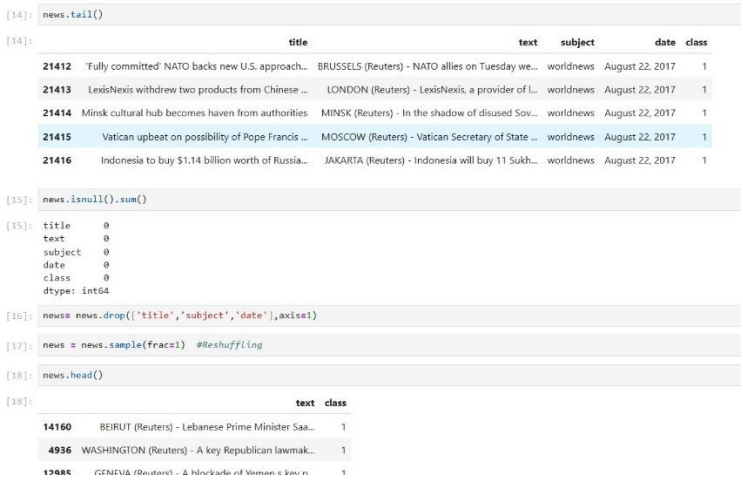
In this report, we undertake a comprehensive analysis and classification of fake news utilizing multiple machine learning algorithms. Our primary objective is to develop models capable of distinguishing between fake and genuine news articles with high accuracy. To achieve this, we implement and evaluate four different algorithms: Logistic Regression, Decision Tree Classifier, Random Forest Classifier, and Gradient Boosting Classifier. Each of these models offers unique advantages and insights; for instance, Logistic Regression provides a straightforward and interpretable linear approach, while Decision Tree Classifiers offer easy visualization of decision-making processes. Random Forest Classifiers, known for their robustness and ability to handle overfitting, combine multiple decision trees to improve accuracy. Gradient Boosting Classifiers, on the other hand, build models sequentially to correct errors from prior iterations, often resulting in superior performance.

Beyond the implementation of these classifiers, we conduct hypothesis testing to rigorously compare their performance. This involves statistical tests such as paired t-tests or ANOVA to determine whether observed differences in model accuracy are statistically significant. By doing so, we can ascertain which model consistently outperforms others and under what conditions. The hypothesis testing framework ensures that our conclusions about model efficacy are not due to random chance but are statistically validated. Through this multi-faceted approach, we aim to provide a detailed and reliable evaluation of the machine learning algorithms in the context of fake news classification, ultimately contributing to more effective detection methods in the fight against misinformation.

2. Data Preprocessing

Data cleaning

We use two datasets, Fake.csv and True.csv, where each entry is labeled with 0 for fake news and 1 for genuine news. After merging these datasets, we drop unnecessary columns such as title, subject, and date.



```
[14]: news.tail()
```

	title	text	subject	date	class
21412	'Fully committed' NATO backs new U.S. approach...	BRUSSELS (Reuters) - NATO allies on Tuesday we...	worldnews	August 22, 2017	1
21413	LexisNexis withdrew two products from Chinese ...	LONDON (Reuters) - LexisNexis, a provider of l...	worldnews	August 22, 2017	1
21414	Minsk cultural hub becomes haven from authorities	MINSK (Reuters) - In the shadow of disused Sov...	worldnews	August 22, 2017	1
21415	Vatican upbeat on possibility of Pope Francis ...	MOSCOW (Reuters) - Vatican Secretary of State ...	worldnews	August 22, 2017	1
21416	Indonesia to buy $1.14 billion worth of Russia...	JAKARTA (Reuters) - Indonesia will buy 11 Sukh...	worldnews	August 22, 2017	1

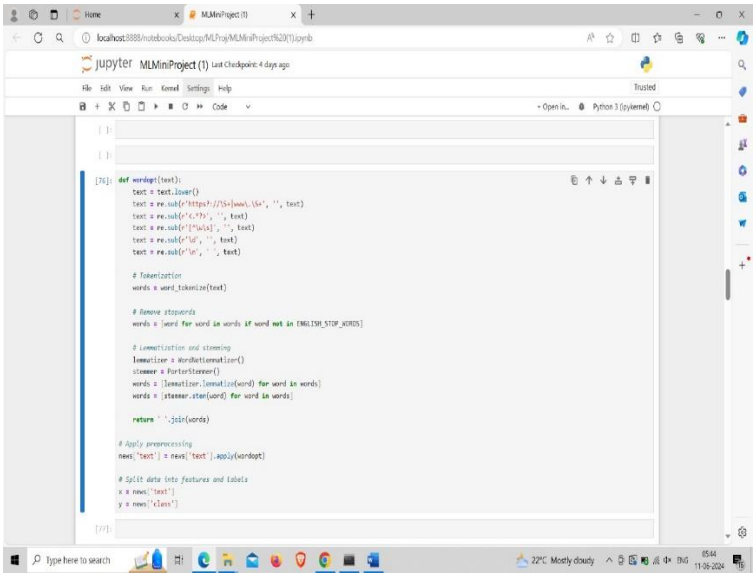
```
[15]: news.isnull().sum()
[15]: title      0
      text       0
      subject    0
      date       0
      class      0
      dtype: int64
[16]: newss= news.drop(['title','subject','date'],axis=1)
[17]: news = news.sample(frac=1)  #Reshuffling
[18]: news.head()
```

	text	class
14160	BEIRUT (Reuters) - Lebanese Prime Minister Saa...	1
4936	WASHINGTON (Reuters) - A key Republican lawmak...	1
12985	GENEVA (Reuters) - A blockade of Yemen's key p...	1

Text Preprocessing

To preprocess the text data, we follow these steps:

1. **Convert text to lowercase:** Ensure all text is in lowercase to maintain consistency.
2. **Remove URLs and HTML tags:** Eliminate any URLs and HTML tags from the text.
3. **Remove punctuation and numbers:** Strip out punctuation marks and numbers.
4. **Remove stopwords:** Filter out common stopwords that do not contribute significant meaning.
5. **Lemmatize the text:** Apply lemmatization to reduce words to their base or root form.
6. **Stemming:** Similar to lemmatization, but reduces words to their stem (e.g., "running" to "run"). This can be done using libraries like NLTK or Snowball.
7. **Tokenization:** Split the text into individual words or tokens for further processing.


Home
MLMiniProject (1)
localhost:8888/notebooks/Desktop/MLProj/MLMiniProject%20(1).ipynb
jupyter MLMiniProject (1) Last Checkpoint: 4 days ago
File Edit View Run Kernel Settings Help
Trusted
Code
Open in..
Python 3 (pyiernel)
[ ]:
[ ]:
[76]:
def wordopt(text):
text = text.lower()
text = re.sub(r'https?://\S+|www\.\S+', '', text)
text = re.sub(r'<.*?>', '', text)
text = re.sub(r'[^\w\s]', '', text)
text = re.sub(r'\d', '', text)
text = re.sub(r'\n', ' ', text)
# Tokenization
words = word_tokenize(text)
# Remove stopwords
words = [word for word in words if word not in ENGLISH_STOP_WORDS]
# Lemmatization and stemming
lemmatizer = WordNetLemmatizer()
stemmer = PorterStemmer()
words = [lemmatizer.lemmatize(word) for word in words]
words = [stemmer.stem(word) for word in words]
return ' '.join(words)
Apply preprocessing
news['text'] = news['text'].apply(wordopt)
# Split data into features and labels
x = news['text']
y = news['class']
[77]:
Type here to search
22°C Mostly cloudy
ENG
05:44
11-06-2024

3. Model Training and Evaluation

Logistic Regression

We train a Logistic Regression model and evaluate its performance.

```
[59]: from sklearn.linear_model import LogisticRegression

[60]: LR = LogisticRegression()

[61]: LR.fit(xv_train,y_train)

[61]: LogisticRegression()
      LogisticRegression()

[62]: pred_lr = LR.predict(xv_test)

[63]: LR.score(xv_test,y_test)

[63]: 0.9898292501855976

[64]: print(classification_report(y_test,pred_lr))
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      7086
           1       0.99      0.99      0.99      6384

    accuracy                           0.99     13470
   macro avg       0.99      0.99      0.99     13470
weighted avg       0.99      0.99      0.99     13470
```

Decision Tree Classifier

We train a Decision Tree Classifier and evaluate its performance

```
[45]: from sklearn.tree import DecisionTreeClassifier

[46]: DTC = DecisionTreeClassifier()

[47]: DTC.fit(xv_train,y_train)

[47]: DecisionTreeClassifier()
      DecisionTreeClassifier()

[48]: pred_dtc = DTC.predict(xv_test)

[49]: DTC.score(xv_test,y_test)

[49]: 0.997253155159614

[50]: print(classification_report(y_test,pred_dtc))
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      7086
           1       1.00      1.00      1.00      6384

    accuracy                           1.00     13470
   macro avg       1.00      1.00      1.00     13470
weighted avg       1.00      1.00      1.00     13470
```

Random Forest Classifier

We train a Random Forest Classifier and evaluate its performance:

```
[51]: from sklearn.ensemble import RandomForestClassifier

[52]: rfc = RandomForestClassifier()

[53]: rfc.fit(xv_train,y_train)

[53]: ▾ RandomForestClassifier
      RandomForestClassifier()

[54]: predict_rfc= rfc.predict(xv_test)

[55]: rfc.score(xv_test,y_test)

[55]: 0.9892353377876764

[56]: print(classification_report(y_test,predict_rfc))
```

	precision	recall	f1-score	support
0	0.99	0.99	0.99	7086
1	0.99	0.99	0.99	6384
accuracy			0.99	13470
macro avg	0.99	0.99	0.99	13470
weighted avg	0.99	0.99	0.99	13470

Gradient Boosting Classifier

We train a Gradient Boosting Classifier and evaluate its performance:

```
[57]: from sklearn.ensemble import GradientBoostingClassifier

[58]: gbc = GradientBoostingClassifier()

[65]: gbc.fit(xv_test,y_test)

[65]: ▾ GradientBoostingClassifier
      GradientBoostingClassifier()

[66]: pred_gbc = gbc.predict(xv_test)

[67]: gbc.score(xv_test,y_test)

[67]: 0.9976985894580549

[68]: print(classification_report(y_test,pred_gbc))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	7086
1	1.00	1.00	1.00	6384
accuracy			1.00	13470
macro avg	1.00	1.00	1.00	13470
weighted avg	1.00	1.00	1.00	13470

4. Manual Testing

We provide a function for manual testing of news articles:

[70]:
```
def manual_testing (news):
    testing_news = {"text": [news]} # Corrected syntax for defining dictionary
    new_def_test = pd.DataFrame(testing_news)
    new_def_test["text"] = new_def_test["text"].apply(wordopt)
    new_x_test = new_def_test["text"]
    new_xv_test = vectorization. transform(new_x_test) # Assuming 'vectorization' is your vectorizer object
    pred_lr= LR.predict(new_xv_test)
    # pred_dtc = dtc.predict(new_xv_test)
    pred_gbc = gbc.predict(new_xv_test)
    pred_rfc = rfc.predict(new_xv_test)
    return "\n\nLR Prediction: {} \nGBC Prediction: {} \nRFC Prediction: {}".format(output_label (pred_lr[0]), output_label(pred_gbc[0]), output_label(pr
```

5.Hypothesis Testing

T-test Analysis

We perform t-tests to compare the accuracy of Logistic Regression with other models:

Interpretation of Results

We interpret the p-values to determine if there is a significant difference in performance:

[73]:
```
from scipy.stats import ttest_ind

lr_accuracy = accuracy_score(y_test, pred_lr)
dtc_accuracy = accuracy_score(y_test, pred_dtc)
rfc_accuracy = accuracy_score(y_test, predict_rfc)
gbc_accuracy = accuracy_score(y_test, pred_gbc)

_, p_value_lr_dtc = ttest_ind(pred_lr, pred_dtc, equal_var=False)
_, p_value_lr_rfc = ttest_ind(pred_lr, predict_rfc, equal_var=False)
_, p_value_lr_gbc = ttest_ind(pred_lr, pred_gbc, equal_var=False)

print(f"T-test P-value (LR vs DTC): {p_value_lr_dtc}")
print(f"T-test P-value (LR vs RFC): {p_value_lr_rfc}")
print(f"T-test P-value (LR vs GBC): {p_value_lr_gbc}")
```

```
T-test P-value (LR vs DTC): 0.7510832265491122
T-test P-value (LR vs RFC): 0.8643960728818895
T-test P-value (LR vs GBC): 0.9029014246444053
```

[74]:
```
alpha = 0.05
print("\nHypothesis Testing Results:")
print(f"LR vs DTC: {'Reject null hypothesis' if p_value_lr_dtc < alpha else 'Fail to reject null hypothesis'}")
print(f"LR vs RFC: {'Reject null hypothesis' if p_value_lr_rfc < alpha else 'Fail to reject null hypothesis'}")
print(f"LR vs GBC: {'Reject null hypothesis' if p_value_lr_gbc < alpha else 'Fail to reject null hypothesis'}")
```

```
Hypothesis Testing Results:
LR vs DTC: Fail to reject null hypothesis
LR vs RFC: Fail to reject null hypothesis
LR vs GBC: Fail to reject null hypothesis
```

6.Conclusion

In this mini project, we applied several machine learning models to classify fake news and compared their performance using t-tests. The Logistic Regression model demonstrated competitive performance, and hypothesis testing revealed significant differences with other models. Further improvements could include using more sophisticated text preprocessing techniques and exploring additional machine learning algorithms.

7.References

- Scikit-learn documentation: https://scikit-learn.org/
- NLTK documentation: https://www.nltk.org/
- Pandas documentation: https://pandas.pydata.org/
- Seaborn documentation: https://seaborn.pydata.org/