Title: A Comparative Study of Machine Learning and Deep Learning Text Classification

Description:

This project explores the performance of various machine learning algorithms in text classification. The main challenge is to classify the textual information according to its content. This is a vital operation for applications such as detecting fake news, spam filtering, and topic categorization.

The project measures the performance of traditional machine learning models (Logistic Regression, Random Forest, Support Vector Machines) with an Artificial Neural Network (ANN) as a baseline. Two major experiments are performed with various datasets and pre-processing methods to understand which combination of model and method best achieves Accuracy and F1-score.

2. Dataset Source

The project is built based on the following two dataset sources however you can find more in the notebook due to laziness level: a) UNESCO Institute of Statistics(January 2005) for the latest by country data on internet usage (%PN°) and b) World Development indicators for bodies such as GDP (ZB104) and Pop for the 2016 7 values.

1. train (2).csv (out of Untitled4.ipynb):

* Data Size: There are 24,353 samples with 4 columns named Unnamed: 0, title, text, and label in this dataset.

Preprocessing:

* The rows that contain any null value was removed with .dropna().

* A mistaken cleansing was performed as a preprocessing step: LabelEncoder() was fitted on the title and text columns. This is a terrible way to treat text, because it replaces text strings with random numbers and loses the semantic meaning of the text.

* The (supposedly) encoded features were then scaled using StandardScaler.

2. Fake.csv (out of Untitled14.ipynb):

* Data Size: Two dataset are loaded here: True.csv and then True.csv is replaced by Fake.csv, so the only dataset used for experiment in this script is "Fake".

Preprocessing:

* A conventional NLP processing method was carried on: TfidfVectorizer was fitted on the text field to extract the text traits as a digital matrix of the TF-IDF features.

* Stop words in English language were removed from the list of ultra frequent words and the vocabulary was trimmed down to 5 000 features.

* The target variable, subject (seems to be multi-class with 6 subjects like 'News', 'politics', etc.), was LabelEncoded to undergo neural network

3. Methods The method adopted in this study is to compare four different classification algorithms to find the best one for the text data in hand.

| Model | Category | Notebook(s) Used In | Description |

| :--- | :--- | :--- | :--- |

| *Logistic Regression* | Linear Model | Both | A baseline linear classifier that works well with high-dimensional sparse data (e.g. TF-IDF). |

| *Random Forest* | Ensemble Model | Both | A tree-based ensemble algorithm that employs bagging to create multiple decision trees and averages the results to improve overfitting. |

| *Support Vector Machine (SVM)* | Kernel Model | Untitled4.ipynb | A very good classifier which searches for an optimal hyperplane. An 'rbf' kernel was employed. |

| *Artificial Neural Network (ANN)* | Deep Learning | Both | A Keras/TensorFlow Sequential model as the deep learning baseline. |

Rationale: This selection covers a broad spectrum of model 'families': linear (LR), ensemble (RF), kernel-based (SVM), and deep learning (ANN). This leads to an investigation of the best method (and the best text preprocessing method) for such a text-based task.

#Humanized Output

4. How to Run the Code

Based on the nbs, the overall flow is:

1. *Import Libraries:* Import the following libraries - pandas, numpy,sklearn(models,metrics,preprocessing) and tensorflow.keras(for the ANN).

2. *Load Data:* Load the pandas DataFrame from the CSV file/(s).

3. *Preprocess Data:*

   * Handle NA's (e.g., data.dropna()).

   * *Select An Approach:*

      * *NLP Approach (NB2):* Use TfidfVectorizer on the text column to create X and use LabelEncoder on the target column to create y.

      * *Wrong Approach (NB1):* Use LabelEncoder and StandardScaler on the text features.

4. *Split Data:* Use train_test_split to split the data into training and testing sets.

5. *Train Classical Models:*

   * Instantiate LogisticRegression, RandomForestClassifier and SVC.

   * Train each model on the training data (X_train, y_train).

6. *Train Neural Network:*

   * Define the architecture of ANN, using Sequential (eg Dense layers) ..

   * Compile the model with an optimizer (adam) and the loss function (binary_crossentropy).

   * Train the ANN on the training set (ann.fit(...)).

7. *Evaluate:*

   * For all models: predict on the X_test.

* Compute metrics - acc, f1, mse, etc.

* Put the results in a DataFrame and print/plot for comparison.

5. Conclusions/Discussion of Results from the Experiments

Two major trials were performed yielding widely diverging results.

Experiment 1: Untitled4.ipynb (LabelEncoder + Scaling)

They applied the defective LabelEncoder technique to the text characteristics to forecast the binary label.

* *Results:*

| Model | Accuracy | F1 Score | MSE |

| :--- | :--- | :--- | :--- |

| Logistic Regression | 0.695 | 0.695 | 0.305 |

| *Random Forest* | *0.951* | *0.951* | *0.049* |

| SVM | 0.771 | 0.763 | 0.229 |

* *On the analysis side, it is worth noting:*

Random Forest also achieved a remarkable accuracy (95.1%). This is *really suspicious* result, given that the pre-processing (which washes out all linguistic information). This suggests that either the model is overfitting or that the label is correlated with some non-textual artifact (like text length or the random number given by the LabelEncoder) instead of the text.

* The ANN was trained but its results *were not successfully compared* in the table with major results. A NameError in cell 27 means that the code to run the ANN evaluation (y_pred_ann = (ann.predict(X_test) > 0.5).astype(int)) was executed before the ANN model (ann = Sequential(...)) in cell 28 was defined and trained.

Experiment 2: Untitled14.ipynb (TF-IDF Vectorizer)

This experiment applied the proper TF- IDF procedure to predict the multi-class subject column on Fake.csv.

* * Results:*

| Model | Accuracy | F1 Score (Macro) | Precision (Macro) |

| — | — | — | — |

| *Logistic Regression* | *0.611* | *0.340* | *0.360* |

| Random Forest | 0.544 | 0.281 | 0.274 |

| ANN | 0.034 | 0.011 | 0.006 |

* *Analysis:*

* Logistic Regression was the best model, however its accuracy (61.1%) and F1-score (0.34) are quite underwhelming for a 6-class problem.

* The *ANN failed catastrophically*, with a mere 3.4% accuracy (a level of performance below random guessing).

* *Why the ANN Failed*: The code has a critical model mismatch. The target variable subject has 6 classes(as per label encoder) but ann was designed with a single output node and sigmoid activation -> Dense(1, activation='sigmoid'). That architecture is for *binary* classification. With a 6-class problem it should be Dense(6, activation=softmax) and loss=sparse_categorical_crossentropy. The ANN's training history (cell 54) shows this exact failure as well - with huge negative loss values, confirming that the loss, output are mismatched in a more fundamental way. ###

6. Conclusion

In this work we illustrate the importance of the choices of the preprocessing and the model architecture in the accuracy of a text classifier.

* *Preprocessing Key Finding 1:* LabelEncoder over raw text (Experiment 1) is a no-go solution, even though it produces high-looking (but probably misleading) scores. The vanilla TfidfVectorizer (Experiment 2) is the proper baseline way to transform text to features that are used by classical ML models.

* *Key Finding 2 (ModelChoice):* A simple Logistic Regression model, gotcha, bested a complicated Random Forest on this task when using sane TF-IDF features.

* *Key Finding 3 (Architecture):* The ANN was a complete disaster in Experiment II, not because ANNs are a poor choice for text, but as a result of a fundamental implementation error . Adopting a binary classification model for a multi-class scenario led to results worse than chance.

Taken together, the moral is that in order for learning to succeed, the whole pipeline needs to be right both from an end-to-end perspective. A simpler model with good preprocessing (LR + TFIDF-based) will always outperform a more complicated model with a bad setup (ANN using an erroneous loss function/wrong number of layers for the task).

References

 Dataset (Untitled14.ipynb):

- Singh, A. (2024). *Fake News Classification* [Dataset]. Kaggle. Retrieved from: https://www.kaggle.com/datasets/aadyasingh55/fake-news-classification/

 Dataset (Untitled4.ipynb):

- *(The source for train (2).csv was not specified, but would be listed here.)*

 Software & Libraries:

- **Pandas:** McKinney, W. (2010). Data structures for statistical computing in Python. *Proceedings of the 9th Python in Science Conference*, 445, 51-56.
- **Scikit-learn:** Pedregosa, F. et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830.
- **TensorFlow/Keras:** Abadi, M. et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

- **Matplotlib:** Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90-95.
- **Seaborn:** Waskom, M. L. (2021). Seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60), 3021.