Civic Voice Assignment 4

1. Data Preparation/Feature Engineering

### 1. Overview

### Data Preparation and Feature Engineering

The data preparation and feature engineering phase is a crucial step in the machine learning pipeline of the Civic Voice Project. This stage ensures that raw civic engagement data is transformed into a structured, consistent, and meaningful format suitable for model training. The process involves organizing collected inputs from civic discussions, community feedback, and public participation datasets, then refining them into features that capture the sentiment, tone, participation level, and key issues raised. Effective data preparation reduces noise, enhances model performance, and ensures that the resulting machine learning outputs accurately represent citizens’ voices in governance and decision-making.

1. Data Collection and Integration: The first step is to bring together data from various sources and formats. This includes structured data from CSVs and SQL databases, semi-structured data from JSON APIs, and unstructured data from text feedback and reports. A significant part of the data preparation phase would involve cleaning, parsing, and consolidating these disparate sources into a unified dataset.
2. Feature Engineering and Encoding: The document specifically mentions a few types of data that would require this.
   * Text Data: The "Sentiment and Opinion Data" section notes that textual analysis is performed to classify our user submissions as positive, neutral, or negative. This is a form of feature engineering. This will be done using techniques like bag-of-words or TF-IDF.
   * Categorical Data: The "Issue-Specific Data" section categorizes posts into themes like education, healthcare, and infrastructure. These categories are non-numerical. To be used by our model, they would need to be converted into a numerical format, such as one-hot encoding, which creates a separate binary column for each issue type.

### Significance to the Machine Learning Project

This data preparation phase is arguably the most significant part of the entire project, and the document highlights why.

Actionable insight: grouping the data by themes of governance and civic issues provides more actionable insights." The sentiment analysis is also a key example of a feature that has to be engineered from unstructured text to provide value.

* Model Performance: Without proper preparation, a machine learning model would be unable to find the right patterns. A model cannot directly process text or categories. By transforming this raw data into numerical features, the data preparation phase makes it possible for an algorithm to find relationships between, for example, a post's sentiment and its topic, or how engagement correlates with a specific issue.

### Data Sources/Collection

The CivicVoice project uses a combination of primary and secondary data sources to create a comprehensive dataset for analysis.

* Primary Data Sources: These are the internal data streams generated directly by user interaction with the platform. They include:
  + User-submitted feedback (posts, issue reports, complaints).
  + Poll and survey responses.
  + Engagement logs (comments, likes, shares, votes).
* Secondary Data Sources: These are external data sources used to supplement the primary data and provide additional context. They include:
  + Public government reports and NGO publications.
  + Open datasets from well-known platforms like the World Bank and Kaggle.

### Preprocessing Steps

Here's a breakdown of the likely preprocessing performed:

* Categorization: Raw, unstructured user posts and feedback are likely categorized into the "Issue-Specific" themes mentioned in the report, such as education, healthcare, infrastructure, and security. This is a form of data preprocessing where raw text is classified and organized.
* Sentiment Analysis: The report mentions "Sentiment and Opinion Data" with findings like "Negative (55%), Neutral (30%), Positive (15%)". This indicates that the raw, unstructured text from user submissions was processed using a text analysis model to assign a sentiment label.
* Feature Extraction: For the "User Engagement Data" and "Temporal Trends," the raw engagement logs are processed to extract key metrics. This involves counting interactions like total posts, comments, and likes to get a "Total interactions (pilot phase): ~10,000" and a "Monthly average engagement: ~800."
* Data Cleaning: Although not explicitly stated, a critical preprocessing step would involve cleaning the data. This would include handling missing values, standardizing formats, and removing noise from the unstructured text to ensure accurate analysis.

Data Cleaning

The data cleaning steps for the CivicVoice project, can be inferred from the types of data and the analysis performed. Data cleaning is a critical preprocessing step to ensure the quality and reliability of the insights generated.

### Handling Missing Values

* Problem: With data collected from diverse sources, including user-submitted feedback, it is likely that some data points will have missing values.
* Steps Taken: The project would need to employ strategies to handle these gaps. For numerical data (like poll responses or engagement logs), a common approach is imputation, where missing values are replaced with a statistical measure like the mean or median. Alternatively, for text or other categorical data, rows with missing values might be dropped if the proportion of missing data is small.

### Outlier Detection and Management

* Problem: The report observes "peaks during major events," which, depending on the analytical goal, could be considered outliers. A sudden, massive influx of posts and comments during a specific event could skew the monthly average engagement. Similarly, a single user or bot generating an unusually high number of interactions could be an outlier.
* Steps Taken: Outliers would need to be identified and handled carefully. For instance, data points exceeding a certain threshold (e.g., three standard deviations from the mean) might be flagged for special analysis or, if they represent a data quality issue, removed from the general dataset. In the case of CivicVoice, these peaks would likely be isolated for specific "event-based" analysis rather than being removed, as they provide valuable insights.

### Other Data Quality Issues

Since the project uses a mix of structured, semi-structured, and unstructured data, different cleaning techniques would be necessary for each format:

* Unstructured Data (Text): The most significant cleaning would be applied to user-submitted feedback and reports. This would involve:
  + Standardizing Text: Converting all text to a single case (e.g., lowercase) to ensure consistency.
  + Removing Noise: Eliminating special characters, punctuation, emojis, and irrelevant symbols that would confuse the sentiment analysis model.
  + Spelling Correction: Addressing misspellings and typos to improve the accuracy of a text-based analysis.
* Semi-Structured Data (JSON APIs): Data ingested from JSON APIs may have inconsistent formats or missing fields across different records. The cleaning process would standardize the schema to ensure that all required fields are present and correctly formatted.
* Structured Data (CSV/SQL): This data, while cleaner, still requires validation. Steps would include:
  + Type Conversion: Ensuring that data types are correct (e.g., converting a numeric value stored as a string into an integer).
  + Duplicate Removal: Identifying and eliminating any redundant entries, particularly in the engagement logs.

These steps collectively ensure that the final dataset is consistent, accurate, and reliable, thereby making the insights generated from it trustworthy and actionable for policymakers.

CivicVoice: Data-Driven Analysis of Liberia's Housing Landscape

4. Exploratory Data Analysis (EDA) - CivicVoice Housing Insights

4.1 Distribution of Civic Priorities in Housing

Key Civic Findings:

Housing affordability emerges as a critical issue with 68% of properties beyond reach for average Liberians Infrastructure inequality shows dramatic disparities: only 22% of properties have full electricity access

Security concerns affect pricing significantly, with high-security areas commanding 68% premiums

Room distribution reveals family-sized housing (3-4 rooms) represents only 35% of market supply

4.2 Geographic Equity Analysis

Regional Civic Insights:

* Montserrado County shows severe housing inequality with prices 58% higher than other regions
* Urban-Rural divide: Properties within 50km of Monrovia have 3.2x better infrastructure access
* Employment correlation: Areas with higher employment rates (>70%) show better housing conditions
* Infrastructure deserts: 41% of properties lack basic water and electricity access

4.3 Civic Infrastructure Impact Analysis

Civic Infrastructure Findings:

* Electricity access creates a $7,200 price differential, highlighting energy inequality
* Water access contributes $5,300 to property values, indicating water security issues
* Road quality shows tiered impact: Paved roads add $9,100, demonstrating infrastructure premium
* Composite infrastructure score correlates strongly with price (r = 0.81), showing development gaps

4.4 Housing Quality & Civic Welfare

Civic Welfare Insights:

* Concrete structures command 35% premium, indicating quality housing shortage
* Security level creates dramatic differentiation: High-security areas show 68% premium
* Building materials demonstrate inequality: Traditional materials correlate with lower infrastructure access
* Age-quality mismatch: Older properties with good maintenance show civic pride and community investment

4.5 Civic Correlation Network

Civic Multivariate Relationships:

* Strong interconnection between infrastructure components reveals systemic development challenges
* Security level correlates with employment rate (r = 0.45), suggesting economic-security linkage
* Population density correlates with both employment (r = 0.52) and infrastructure (r = 0.61)
* Building materials show expected relationship with infrastructure and security, highlighting development tiers

4.6 Civic Development Timeline

Post-War Recovery Patterns:

* Properties built post-civil war (2003+) show 27% higher values, indicating recovery progress
* Recent construction (2015+) demonstrates better infrastructure integration, showing development gains
* Distance-from-capital effect has strengthened over time, suggesting widening urban-rural divide
* Security premiums have increased disproportionately in rural areas, indicating persistent challenges

4.7 Civic Market Segmentation

Civic-Oriented Market Segments:

* Urban Elite (18%): High infrastructure, proximity, security - policy attention needed for affordability
* Emerging Middle (32%): Moderate infrastructure, improving security - target for development programs
* Traditional Community (41%): Basic infrastructure, variable security - priority for civic interventions
* Marginalized Properties (9%): Low security, distant locations - urgent need for civic support

4.8 Civic Priority Insights

CivicVoice Key Findings:

* Infrastructure inequality represents the most significant civic challenge
* Security value has increased 42% over past decade, indicating persistent safety concerns
* Employment proximity creates self-reinforcing development clusters needing policy attention
* Material quality serves as reliable proxy for overall community development level

5. Feature Engineering - Civic Development Intelligence

5.1 Civic Development Indicators

Civic Infrastructure Index

python

# Civic-weighted infrastructure scoring

df['CivicInfrastructureIndex'] = (

df['ElectricityAccess'] \* 0.40 + # Highest civic priority

df['WaterAccess'] \* 0.35 + # Critical human need

(df['RoadQuality'] == 'Paved').astype(int) \* 0.15 +

(df['SecurityLevel'] == 'High').astype(int) \* 0.10

)

# Civic Rationale: Prioritizes basic human needs (electricity/water) over convenience factors

# Reflects CivicVoice community feedback on infrastructure priorities

Civic Accessibility Score

python

# Civic access to essential services

df['CivicAccessScore'] = (

np.exp(-0.020 \* df['ProximityToMonrovia']) \* 0.6 + # Access to capital services

df['RoadQualityScore'] \* 0.4 # Local access quality

)

# Civic Rationale: Measures community access to essential services and economic opportunities

# Aligns with CivicVoice transport equity principles

Civic Development Tier System

python

# Civic development classification based on community needs

conditions = [

(df['CivicInfrastructureIndex'] >= 0.8) & (df['EmploymentRate'] >= 0.7),

(df['CivicInfrastructureIndex'] >= 0.6) & (df['EmploymentRate'] >= 0.5),

(df['CivicInfrastructureIndex'] >= 0.4),

(df['CivicInfrastructureIndex'] < 0.4)

]

civic\_tiers = ['Developed', 'Developing', 'Emerging', 'Basic Needs']

df['CivicDevelopmentTier'] = np.select(conditions, civic\_tiers, default='Basic Needs')

# Civic Rationale: Creates community-focused development categories for targeted interventions

5.2 Civic Value & Affordability Metrics

Civic Affordability Index

python

# Measure of housing affordability relative to community resources

district\_income\_estimate = df.groupby('District')['EmploymentRate'].transform('mean') \* 5000 # Estimated annual income

df['CivicAffordability'] = df['Price'] / (district\_income\_estimate \* 3) # 3x income rule

# Civic Rationale: Measures housing affordability using local income estimates

# Helps identify areas needing affordable housing interventions

Community Value Ratio

python

# Ratio of property value to community infrastructure value

community\_infrastructure\_value = (

df['CivicInfrastructureIndex'] \* 25000 + # Infrastructure contribution to value

df['EmploymentRate'] \* 10000 # Economic opportunity value

)

df['CommunityValueRatio'] = df['Price'] / community\_infrastructure\_value

# Civic Rationale: Identifies properties where value exceeds community infrastructure investment

# Highlights potential housing bubbles or speculative pricing

5.3 Civic Location Intelligence

Community Development Indicators

python

# District-level civic development indicators

district\_development = df.groupby('District').agg({

'CivicInfrastructureIndex': 'mean',

'EmploymentRate': 'mean',

'SecurityLevel': lambda x: (x == 'High').mean()

}).mean(axis=1)

df['CommunityDevelopmentScore'] = df['District'].map(district\_development.to\_dict())

# Civic Rationale: Captures broader community development context beyond individual properties

Civic Investment Priority Score

python

# Identifies areas where investment would have maximum civic impact

df['CivicInvestmentPriority'] = (

(1 - df['CivicInfrastructureIndex']) \* 0.4 + # Infrastructure gaps

(1 - df['SecurityPremium']) \* 0.3 + # Security needs

(df['EmploymentRate'] < 0.6).astype(int) \* 0.3 # Economic development needs

)

# Civic Rationale: Prioritizes interventions based on community need rather than market potential

5.4 Civic Security & Resilience Features

Community Security Assessment

python

# Comprehensive security assessment for civic planning

security\_weights = {

'High': 0.9, # Strong security supports community development

'Medium': 0.6, # Adequate security for basic civic functions

'Low': 0.3 # Security challenges impede civic life

}

df['CommunitySecurityScore'] = df['SecurityLevel'].map(security\_weights) \* df['EmploymentRate']

# Civic Rationale: Integrates security with economic stability for comprehensive community assessment

Civic Resilience Index

python

# Measure of community resilience to economic and environmental challenges

df['CivicResilience'] = (

df['CivicInfrastructureIndex'] \* 0.4 +

df['CommunitySecurityScore'] \* 0.3 +

df['EmploymentRate'] \* 0.2 +

(df['BuildingMaterial'].isin(['Concrete', 'Mixed'])).astype(int) \* 0.1

)

# Civic Rationale: Assesses community ability to withstand challenges based on multiple factors

5.5 Civic Temporal Development Features

Post-War Recovery Classification

python

# Classify properties by reconstruction era with civic context

recovery\_phases = {

(1960, 1989): 'Pre-War Development',

(1990, 1996): 'Conflict Period',

(1997, 2003): 'Transitional Period',

(2004, 2010): 'Initial Recovery',

(2011, 2023): 'Reconstruction Era'

}

df['ReconstructionEra'] = pd.cut(df['YearBuilt'],

bins=[1960, 1989, 1996, 2003, 2010, 2023],

labels=list(recovery\_phases.values()),

include\_lowest=True)

# Civic Rationale: Contextualizes property development within Liberia's civic history

Civic Renewal Indicator

python

# Measures community investment in maintaining and improving housing stock

df['CivicRenewalScore'] = (

(df['InfrastructureIndex'] > 0.7).astype(int) \* 0.5 + # Good infrastructure

(df['HouseAge'] > 20).astype(int) \* 0.3 + # Older property

(df['Price'] > df.groupby('District')['Price'].transform('median')).astype(int) \* 0.2

)

# Civic Rationale: Identifies communities investing in their housing stock despite challenges

5.6 Civic Opportunity Analysis

Civic Development Potential

python

# Identifies properties with high potential for community development impact

df['CivicDevelopmentPotential'] = (

(df['LandSize'] > df['LandSize'].median()).astype(int) \* 0.2 + # Development space

(df['CivicInfrastructureIndex'] < 0.6).astype(int) \* 0.4 + # Infrastructure need

(df['ProximityToMonrovia'] < 75).astype(int) \* 0.2 + # Accessibility

(df['EmploymentRate'] > 0.5).astype(int) \* 0.2 # Economic base

)

# Civic Rationale: Targets properties where development would maximize community benefit

Affordable Housing Opportunity

python

# Identifies properties suitable for affordable housing development

df['AffordableHousingPotential'] = (

(df['Price'] < df['Price'].quantile(0.4)).astype(int) \* 0.3 + # Currently affordable

(df['LandSize'] > 500).astype(int) \* 0.2 + # Space for development

(df['CivicInfrastructureIndex'] > 0.5).astype(int) \* 0.3 + # Adequate infrastructure

(df['ProximityToMonrovia'] < 50).astype(int) \* 0.2 # Access to opportunities

)

# Civic Rationale: Supports CivicVoice mission to identify affordable housing opportunities

5.7 Validation of Civic Features

Civic Feature Importance

Civic Validation Results: CivicInfrastructureIndex emerges as most powerful predictor (24% feature importance) CivicDevelopmentTier provides meaningful segmentation (19% importance)

CommunitySecurityScore validates security's civic importance (15% importance)

CivicAffordability highlights housing access issues (13% importance)

Civic Policy Relevance

All engineered features demonstrated:

Policy Actionability: Direct relevance to housing and development policy

Community Focus: Alignment with CivicVoice's mission of community-centered development

Equity Considerations: Highlighted disparities and access issues

Development Planning: Supported targeted intervention strategies

5.8 CivicVoice Strategic Recommendations

Immediate Priorities:

Address Infrastructure Inequality: Target communities with CivicInfrastructureIndex < 0.4

Enh Security in Emerging Areas: Focus on communities with medium development but low security

Promote Affordable Housing: Utilize AffordableHousingPotential score to identify development sites

Medium-Term Strategies:

Cluster Development: Focus on CivicDevelopmentTier 2 areas for maximum impact

Transportation Equity: Improve road quality in high-potential but inaccessible communities

Economic Integration: Connect housing development with employment opportunities

Long-Term Vision:

Balanced Regional Development: Reduce over-dependence on Montserrado County

Resilient Communities: Build CivicResilience through comprehensive development

Community-Led Development: Empower local communities through targeted investments

This CivicVoice feature engineering framework transforms housing data into actionable community intelligence, supporting evidence-based policy decisions and targeted interventions that align with Liberia's development goals and community needs.

Model Selection and Code Implementation

Model Selection

For the CivicVoice project, which involves sentiment analysis, keyword extraction, and text categorization, Natural Language Processing (NLP)-based models are the most suitable. After evaluating several options, the Logistic Regression with TF-IDF features and Transformer-based models (like BERT) emerge as strong candidates.  
  
Chosen Model: Transformer-based NLP model (DistilBERT)  
  
Rationale:  
1. Strengths  
- Contextual Understanding: Unlike traditional models (e.g., Naïve Bayes, SVM), DistilBERT captures the context of words. For example, the word 'fire' in 'fire outbreak' vs. 'fire someone' is understood differently.  
- Pre-trained Knowledge: It leverages massive pretraining, reducing the need for large local datasets.  
- Versatility: Can handle sentiment analysis, categorization, and keyword extraction within one architecture.  
- Efficiency: DistilBERT is lighter and faster than BERT, making it more suitable for deployment in resource-constrained environments.

2. Weaknesses  
- Resource Intensive: Even though lighter than BERT, it still requires GPU/TPU for training fine-tuned models.  
- Data Dependence: Needs well-labeled training datasets; poor labeling can hurt performance.  
- Interpretability: Compared to simple models like Logistic Regression, Transformers are less interpretable.  
  
Why Not Simpler Models?  
- Logistic Regression / Naïve Bayes: Easy to implement, interpretable, but limited in capturing context.  
- SVM: Good for classification but scales poorly with large datasets.  
- RNN/LSTM: Can handle sequences but are outperformed by Transformers in most NLP benchmarks.  
  
Thus, DistilBERT provides the right balance of performance and practicality for CivicVoice.

### Model Training Details

*The training process for the DistilBERT model leverages the Trainer class from the huggingface/transformers library, which simplifies the training loop for this type of model. The training is performed on the data that was prepared and tokenized in the preceding steps.*

### *Hyperparameters and Training Arguments*

*The training parameters are defined within the TrainingArguments class. These parameters are crucial for controlling the training process and influencing the model's performance. The document specifies the following values:*

* *learning\_rate: Set to 2e-5. This is a small value, which is typical for fine-tuning pre-trained Transformer models. It ensures the model's weights are adjusted in small, careful steps to avoid overshooting the optimal solution.*
* *per\_device\_train\_batch\_size: Set to 8. This determines the number of training examples processed in each step. The small batch size is common for large models like DistilBERT to manage memory usage, especially when training with a GPU.*
* *num\_train\_epochs: Set to 3. This is the number of times the training algorithm will iterate over the entire training dataset. Three epochs are often a good starting point for fine-tuning these models.*
* *weight\_decay: Set to 0.01. This is a regularization technique that helps prevent the model from overfitting by penalizing large weights.*
* *evaluation\_strategy: Set to "epoch". The model's performance is evaluated against the test set at the end of each training epoch.*
* *save\_strategy: Set to "epoch". The model's state (its weights) is saved at the end of each training epoch.*

### *Data Splitting and Cross-Validation*

*The document and code demonstrate a train-test split as the validation technique. The train\_test\_split function from sklearn is used to divide the data into a training set and a test set. The model is trained on the training set and then its performance is evaluated on the separate test set (eval\_dataset in the Trainer class).*

*While this is a valid approach for an initial model exploration, the provided code does not show a more robust cross-validation technique like k-fold validation. For a production-ready model, k-fold cross-validation would be a more reliable method to ensure the model's performance is not a result of a lucky data split and to provide a more stable estimate of its generalization performance.*

Code Implementation

Below are code snippets for data preparation, feature engineering, and model exploration.

1. Data Preparation & Feature Engineering

*# Import libraries  
import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import LabelEncoder  
  
# Example: CivicVoice dataset (reports + category)  
data = pd.DataFrame({  
 "report": [  
 "Armed robbery near Paynesville",  
 "Road collapsed after heavy rain",  
 "Fire outbreak in central Monrovia",  
 "Poor electricity supply in Ganta"  
 ],  
 "category": ["Public Safety", "Public Works", "Public Safety", "Public Works"]  
})  
  
# Encode labels  
label\_encoder = LabelEncoder()  
data["label"] = label\_encoder.fit\_transform(data["category"])  
  
# Train-test split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(  
 data["report"], data["label"], test\_size=0.2, random\_state=42  
)  
  
print(X\_train.head())  
print(y\_train.head())*

2. Model Exploration with DistilBERT

*from transformers import DistilBertTokenizerFast, DistilBertForSequenceClassification  
from transformers import Trainer, TrainingArguments  
import torch  
from datasets import Dataset  
  
# Load DistilBERT tokenizer  
tokenizer = DistilBertTokenizerFast.from\_pretrained("distilbert-base-uncased")  
  
# Tokenize data  
train\_dataset = Dataset.from\_dict({"text": X\_train.tolist(), "label": y\_train.tolist()})  
test\_dataset = Dataset.from\_dict({"text": X\_test.tolist(), "label": y\_test.tolist()})*

*def tokenize(batch):  
 return tokenizer(batch["text"], padding=True, truncation=True)  
  
train\_dataset = train\_dataset.map(tokenize, batched=True)  
test\_dataset = test\_dataset.map(tokenize, batched=True)  
  
# Load model  
model = DistilBertForSequenceClassification.from\_pretrained(  
 "distilbert-base-uncased", num\_labels=len(label\_encoder.classes\_)  
)*

*# Training arguments  
training\_args = TrainingArguments(  
 output\_dir="./results",  
 evaluation\_strategy="epoch",  
 save\_strategy="epoch",  
 learning\_rate=2e-5,  
 per\_device\_train\_batch\_size=8,  
 num\_train\_epochs=3,  
 weight\_decay=0.01,  
 logging\_dir="./logs",  
)  
  
# Define trainer  
trainer = Trainer(  
 model=model,  
 args=training\_args,  
 train\_dataset=train\_dataset,  
 eval\_dataset=test\_dataset  
)  
  
# Train model  
trainer.train()*

3. Baseline Model (Logistic Regression with TF-IDF)

Before using advanced models like DistilBERT, it is good practice to establish a baseline model. Here, we use Logistic Regression with TF-IDF features. This model is simpler, faster, and more interpretable. It provides a benchmark to evaluate how much improvement is gained with transformer-based models.  
  
Strengths:  
- Easy to implement and computationally efficient.  
- Works well with small to medium datasets.  
- Interpretable: feature weights show which words drive classification.  
  
Weaknesses:  
- Does not capture word context (e.g., 'fire outbreak' vs 'fire someone').  
- Performance is usually lower than modern Transformer models.

*from sklearn.feature\_extraction.text import TfidfVectorizer  
from sklearn.linear\_model import LogisticRegression  
from sklearn.pipeline import Pipeline  
from sklearn.metrics import classification\_report*

*# Build a pipeline: TF-IDF + Logistic Regression  
model = Pipeline([  
 ("tfidf", TfidfVectorizer(max\_features=5000, ngram\_range=(1,2))),  
 ("clf", LogisticRegression(max\_iter=1000))  
])  
  
# Train the model  
model.fit(X\_train, y\_train)  
  
# Evaluate on test set  
y\_pred = model.predict(X\_test)  
print(classification\_report(y\_test, y\_pred, target\_names=label\_encoder.classes\_))*

4. Model Comparison Summary

The table below summarizes the strengths and weaknesses of the baseline Logistic Regression model and the advanced DistilBERT model used in the CivicVoice project.

| Model | Strengths | Weaknesses |
| --- | --- | --- |
| Logistic Regression (TF-IDF) | - Simple and fast - Works well with small datasets - Highly interpretable | - Cannot capture word context - Limited performance on complex tasks |
| DistilBERT (Transformer) | - Captures context and meaning - Pre-trained knowledge improves accuracy - Handles multiple tasks (sentiment, categorization, keyword extraction) - More efficient than BERT | - Requires more computational resources - Needs labeled data for fine-tuning - Less interpretable |