Data Scientist Technical Test Preparation Guide

Dataset Overview: Fake vs Real News Classification

You'll be working with a **text classification problem** to distinguish between fake and real news articles. This is a binary classification task that will test your NLP and ML skills.

1. Environment Setup (Do This First)

Essential Libraries to Install

bash

```bash
pip install pandas numpy scikit-learn xgboost matplotlib seaborn
pip install nltk textblob wordcloud plotly
pip install jupyter notebook  # if you prefer notebooks
```

IDE Setup

- Ensure your Visual Studio Code has Python extension installed
- Test that you can run Python scripts and see outputs
- Familiarize yourself with debugging tools

2. Dataset Exploration & Analysis

Key Areas to Master

Data Loading & Initial Inspection

- Load CSV files efficiently with pandas
- Check data shape, column names, data types

- Identify missing values, duplicates
- Understand the target variable distribution

Text Data Analysis

- Article length distributions (word count, character count)
- Vocabulary analysis (unique words, common words)
- Text quality assessment (special characters, formatting issues)
- Class imbalance analysis

Visualization Skills

- Distribution plots (histograms, box plots)
- Word clouds for different classes
- Text length comparisons between fake/real news
- Correlation heatmaps for numerical features

3. Feature Engineering (Critical Section)

Text Preprocessing Pipeline

- **Text cleaning**: Remove HTML tags, special characters, URLs
- **Normalization**: Lowercase conversion, whitespace handling
- **Tokenization**: Split text into words/tokens
- **Stop word removal**: Remove common words (the, and, is, etc.)
- **Stemming/Lemmatization**: Reduce words to root forms

Feature Extraction Methods

- **Bag of Words (BoW)**: CountVectorizer
- **TF-IDF**: Term Frequency-Inverse Document Frequency
- **N-grams**: Unigrams, bigrams, trigrams
- **Text statistics**: Length, punctuation count, capitalization ratio
- **Readability scores**: If time permits

Advanced Features (Bonus Points)

- Named Entity Recognition (NER) features
- Sentiment analysis scores
- POS (Part of Speech) tag distributions
- Topic modeling features (LDA)

4. Model Selection & Implementation

Baseline Models (Start Here)

- **Logistic Regression**: Simple, interpretable
- **Naive Bayes**: Works well with text data
- **Random Forest**: Good baseline for any problem

Enhanced Models

- **XGBoost**: Gradient boosting, excellent performance
- **Support Vector Machine (SVM)**: Good for text classification
- **Neural Networks**: If you're comfortable (MLPClassifier)

Implementation Strategy

1. Start with simple TF-IDF + Logistic Regression baseline
2. Gradually add complexity (feature engineering, better models)
3. Compare performance systematically

5. Model Evaluation & Performance Analysis

Evaluation Metrics

- **Accuracy**: Overall correctness
- **Precision**: True positives / (True positives + False positives)
- **Recall**: True positives / (True positives + False negatives)
- **F1-Score**: Harmonic mean of precision and recall
- **ROC-AUC**: Area under ROC curve
- **Confusion Matrix**: Detailed error analysis

Cross-Validation

- Use stratified k-fold cross-validation
- Ensure consistent evaluation across models
- Report mean and standard deviation of metrics

6. Feature Importance Analysis

Methods to Master

- **Coefficients**: For linear models (Logistic Regression)
- **Feature Importance**: For tree-based models (Random Forest, XGBoost)
- **Permutation Importance**: Model-agnostic approach

- **SHAP values**: Advanced interpretability (bonus)

Visualization

- Bar plots of top important features
- Word clouds of important terms
- Feature importance heatmaps

7. Results Visualization & Communication

Key Visualizations

- Model performance comparison charts
- ROC curves for different models
- Feature importance plots
- Confusion matrices with heatmaps
- Learning curves (training vs validation performance)

Communication Skills

- Clear methodology explanations
- Justify your feature engineering choices
- Explain why certain models performed better
- Discuss potential improvements and limitations

8. Technical Implementation Checklist

Code Quality

- Write clean, readable code with comments

- Use functions for reusable code blocks
- Handle errors gracefully (try-catch blocks)
- Follow PEP 8 style guidelines

Performance Considerations

- Use vectorized operations (pandas/numpy)
- Consider memory usage with large datasets
- Time your model training and prediction phases
- Use appropriate data types (category for categorical variables)

9. Common Pitfalls to Avoid

Data Leakage

- Don't use future information to predict past events
- Ensure proper train-test split before any preprocessing
- Be careful with feature scaling and encoding

Overfitting

- Use cross-validation consistently
- Monitor training vs validation performance
- Apply regularization when appropriate

Text Processing Mistakes

- Don't remove too much information during cleaning
- Be consistent with preprocessing across train/test sets

- Handle edge cases (empty strings, special characters)

10. Time Management Strategy

Hour 1: Data Exploration

- Load data and understand structure
- Basic EDA and visualizations
- Identify data quality issues

Hour 2: Feature Engineering

- Text preprocessing pipeline
- Create TF-IDF features
- Generate additional text-based features

Hour 3: Baseline Modeling

- Implement 2-3 baseline models
- Establish evaluation framework
- Get initial performance metrics

Hour 4: Model Enhancement

- Try advanced models (XGBoost)
- Feature selection/engineering refinements
- Hyperparameter tuning (if time allows)

Hour 5: Analysis & Presentation

- Feature importance analysis

- Create compelling visualizations
- Prepare methodology explanations

11. Quick Reference Code Snippets

Data Loading

```python
import pandas as pd
df = pd.read_csv('news_data.csv')
print(df.shape, df.columns.tolist())
print(df['label'].value_counts())
```

Basic Text Preprocessing

```python
import re
from sklearn.feature_extraction.text import TfidfVectorizer

def clean_text(text):
    text = text.lower()
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    return text

# Apply preprocessing
df['cleaned_text'] = df['text'].apply(clean_text)
```

Model Training Template

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=

# Train model
model = LogisticRegression()
model.fit(X_train, y_train)

# Evaluate
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
```

12. Final Tips

- **Start simple**: Get a working pipeline first, then enhance
- **Document everything**: Comment your code and reasoning
- **Think creatively**: Novel features or approaches will stand out
- **Balance performance with interpretability**: Explain your model choices
- **Practice explaining**: Be ready to walk through your methodology
- **Stay calm**: Focus on demonstrating your systematic approach

Key Success Factors

1. **Systematic approach**: Follow the ML pipeline methodically
2. **Strong EDA**: Show deep understanding of the data

3. **Creative features**: Go beyond basic TF-IDF
4. **Model comparison**: Don't just use one model
5. **Clear communication**: Explain your decisions well
6. **Practical insights**: What would you do differently in production?

Good luck with your technical test! Remember, they want to see your thought process and methodology as much as your final results.