This example will use a dataset of customer reviews for a fictional e-commerce platform, combining structured data (user information, product details) with unstructured data (review text).

This comprehensive pipeline covers all the requested components:

1. Data Ingestion and Storage: We use TinyDB to store simulated customer review data.
2. Data Processing and Analysis: Pandas is used for data cleaning and feature engineering.
3. Data Visualization: We create both static (Matplotlib/Seaborn) and interactive (Plotly) visualizations.
4. Statistical Analysis: Statsmodels is used to perform a simple regression analysis.
5. Machine Learning with PySpark: We use Spark MLlib to train a Random Forest Classifier.
6. Deep Learning with TensorFlow/Keras: A simple neural network is built to predict ratings.
7. NLP with NLTK and Transformers: We use NLTK for text preprocessing and BERT for sentiment analysis.
8. Model Deployment: A basic Flask app structure is provided for deployment (though not fully implemented for brevity).

This pipeline demonstrates the integration of various data science and machine learning techniques on a simulated e-commerce customer review dataset. It showcases data handling, analysis, visualization, and multiple modeling approaches including traditional machine learning, deep learning, and NLP.

This Streamlit app does the following:

1. It creates a user interface where users can input customer information and review text.
2. When the "Analyze Review" button is clicked, it uses the deep learning model to predict the rating based on customer age, review word count, and purchase amount.
3. It also uses the BERT model to perform sentiment analysis on the review text.
4. The results are then displayed to the user.

Note that this example assumes you have already trained and saved your models. You'll need to adjust the model loading code if your model saving method is different.

Also, remember that loading large models like BERT directly in a Streamlit app can be memory-intensive and slow. For a production environment, you might want to consider serving these models separately (e.g., using TensorFlow Serving) and having your Streamlit app call an API endpoint instead.

This Streamlit app provides a simple, interactive way for users to input data and get predictions from your models. You can further enhance it by adding more visualizations, explanations of the results, or additional features as needed.