Data Science & AI Project Portfolio

Crafted by www.learnwithbobby.com

Real-World Projects Using Python, ML, NLP & Deep Learning

A curated collection of 10 practical, resume-worthy projects designed to showcase skills in data analysis, machine learning, computer vision, natural language processing, and deployment.

What You'll Find Inside:

- ✅ Project Titles with Descriptions
- ⚙️ Tools & Technologies Used
- 🔁 Step-by-step Workflow
- 🎯 Final Project Outputs

Suitable For:

- Data Science & AI Beginners to Intermediate Learners
- Job Seekers looking to build a practical portfolio
- Students & Developers wanting to showcase real-world skills

🗓️ Last Updated: July 2025
🌐 Website: www.learnwithbobby.com

1. YouTube Video Analysis & Insight Generator

⚙️ **Tools:** Python, YouTube Data API, Pandas, Matplotlib, Streamlit

📄 **Description:**
Analyze YouTube channel or video statistics like views, likes, comments, and frequency.
Get insights into content performance and user engagement trends.
Ideal for creators and marketers optimizing video strategy.

🔁 **Workflow:**

1. Authenticate with YouTube Data API
2. Fetch video/channel metadata
3. Parse JSON response into DataFrame
4. Clean and process metrics
5. Create visual summaries (line/bar plots)
6. Deploy dashboard using Streamlit
7. Allow channel ID/video ID as input

🎯 **Final Output:**
A dashboard showing video frequency, avg views, like-to-comment ratio, etc.
Users can input any channel ID to view insights.
Great for understanding audience trends over time.

---

2. Stock Price Trend Prediction

⚙️ **Tools:** Python, yfinance, Pandas, Scikit-learn, XGBoost, Streamlit

📄 **Description:**
Forecast stock prices using historical financial data and machine learning.
Helps users visualize price trends and make informed decisions.
Great project for ML time series beginners.

🔁 **Workflow:**

1. Fetch stock data using yfinance
2. Create lag features and moving averages
3. Split dataset into train/test
4. Train model (e.g., XGBoost or Linear Regression)

5. Evaluate prediction accuracy
6. Visualize actual vs predicted prices
7. Build Streamlit app for user input

🎯 **Final Output:**
A web app that predicts future stock price for any ticker.
Shows plots for historical vs predicted trends.
Interactive dropdown for choosing companies.

---

3. Resume Job Role Classifier

⚙️ **Tools:** Python, Pandas, NLTK, Scikit-learn, TF-IDF, Flask

📄 **Description:**
Classify resumes into roles like "Data Scientist" or "Web Developer" using NLP.
Useful for HR teams to automate resume screening.
Trains on labeled job/resume datasets.

🔁 **Workflow:**

1. Load resume text data and job labels
2. Clean text using NLTK (stopwords, tokenization)
3. Extract TF-IDF features
4. Train a classifier (e.g., Logistic Regression or SVM)
5. Build Flask file upload interface
6. Accept resume files and extract text
7. Predict and display job role output

🎯 **Final Output:**
User uploads resume → output shows predicted role (e.g., "ML Engineer").
Also includes confidence level or top 3 possible roles.
Helps recruiters shortlist faster.

4. Fake News Detection System

⚙️ **Tools:** Python, Pandas, NLTK, Scikit-learn, TF-IDF, Flask

📄 **Description:**
Detect fake news headlines or articles using text classification.
Uses NLP preprocessing and machine learning techniques.
Helpful in moderating misinformation online.

🔁 **Workflow:**

1. Load dataset of real and fake news
2. Clean text (lemmatization, stopword removal)
3. Vectorize using TF-IDF
4. Train classifier (e.g., Naive Bayes or SVM)
5. Evaluate accuracy and F1-score
6. Build Flask web input interface
7. Return “Fake” or “Real” label

🎯 **Final Output:**
User pastes an article or headline → gets classification result.
Model shows "Fake" or "Real" with confidence score.
Simple interface for testing real-world news.

---

5. Sign Language Detection to Text & Speech

⚙️ **Tools:** Python, OpenCV, TensorFlow/Keras, pyttsx3

📄 **Description:**
Recognizes ASL alphabet gestures from webcam in real time.
Converts gestures into text and speaks them aloud.
Bridges communication for hearing-impaired users.

🔁 **Workflow:**

1. Train CNN model on ASL alphabet dataset
2. Use OpenCV to capture real-time hand gestures
3. Preprocess input frames
4. Predict alphabet for each gesture

5. Store letters in sequence to form word/sentence
6. Convert final text to speech using pyttsx3
7. Display + speak the output

🎯 **Final Output:**
User performs ASL letters in front of webcam.
App builds and displays corresponding text and speaks it aloud.
Useful for accessibility, communication, and education.

---

6. Mental Health Chat Sentiment Analyzer

⚙️ **Tools:** Python, VADER, TextBlob, Streamlit

📄 **Description:**
Analyzes emotional tone of user chat input to detect risk signals.
Flags negative or concerning messages in real time.
Used in mental health bots or support apps.

🔁 **Workflow:**

1. Accept user-entered text
2. Clean and normalize the message
3. Analyze sentiment using VADER or TextBlob
4. Extract polarity and subjectivity scores
5. Display label (positive/neutral/negative)
6. Flag if text has high negativity
7. Show emotional heatmap over time (optional)

🎯 **Final Output:**
App analyzes message and returns emotional label.
Shows live feedback on user mood over time.
Alerts in case of depression, anxiety-related content.

7. Image Classification with CNN (Cats vs Dogs)

⚙️ **Tools:** Python, TensorFlow/Keras, OpenCV, Streamlit

📄 **Description:**
Train a CNN model to classify animal images as “Cat” or “Dog”.
Deploy it with a simple upload-based interface.
Teaches basics of computer vision and classification.

🔁 **Workflow:**

1. Load and preprocess images (resize, normalize)
2. Build and train CNN using Keras
3. Save and load the trained model
4. Create UI for image upload
5. Process uploaded image with OpenCV
6. Run inference and get class
7. Display predicted label with image

🎯 **Final Output:**
Upload image → model classifies it as Cat or Dog.
Prediction is shown with confidence level.
Great for visual model demonstration.

---

8. Face Emotion Recognition

⚙️ **Tools:** Python, OpenCV, DeepFace or Keras, Streamlit

📄 **Description:**
Detect and classify facial emotions like Happy, Sad, Angry in real time.
Uses webcam and deep learning models for expression analysis.
Applies to HCI, emotion tracking, and smart interfaces.

🔁 **Workflow:**

1. Use OpenCV to capture face frames
2. Detect and extract facial ROI
3. Feed face into emotion classifier
4. Predict emotion class (e.g., Happy, Neutral)

5. Overlay result on video feed
6. Stream results in real time
7. Build web-based demo UI

🎯 **Final Output:**
Webcam feed with real-time emotion label overlay.
User expressions are classified instantly.
Useful in UI/UX testing and emotion-driven applications.

---

9. AI-Based Recipe Generator from Ingredients

⚙️ **Tools:** Python, Hugging Face Transformers (T5/BART), NLTK, Streamlit, OpenAI API (optional)

📄 **Description:**
Takes a list of ingredients and generates a full recipe using NLP.
Provides a recipe title, steps, and cook time or calories.
Helps users cook with whatever they have at home.

🔁 **Workflow:**

1. Accept input list of ingredients
2. Format into prompt for text generation
3. Use fine-tuned T5/BART or GPT-3 to generate recipe
4. Extract and format: title, steps, cook time
5. Display recipe in readable format
6. Add emojis, images or categories (optional)
7. Deploy with Streamlit for easy access

🎯 **Final Output:**
User inputs “eggs, cheese, tomato” → App shows omelet recipe.
Includes step-by-step guide and estimated cook time.
Looks like a cooking site’s auto-suggested recipe.

---

10. AI Comment Classifier (Toxic Comment Detection)

⚙️ **Tools:** Python, TensorFlow/Keras, Scikit-learn, NLTK, Streamlit

📄 **Description:**
Classify online comments into categories like “Insult”, “Threat”, “Love”, “Hate”, “Harassment”.

Trains a multi-label classification model.
Useful for social media moderation.

🔁 **Workflow:**

1. Load and clean comment text
2. Use NLTK for tokenization and stemming
3. Vectorize using TF-IDF or embeddings
4. Train a multi-label classifier
5. Accept user comment input
6. Predict label(s) from trained model
7. Display result using Streamlit

🎯 **Final Output:**
Comment: “You’re disgusting and dumb” → Output: ["Insult", "Hate"]
Also supports positive tags like ["Love", "Support"].
Great for moderating online communities or feedback tools.