

Week 1 Progress Report – Machine Learning Foundations

Days 1–2: NumPy & Pandas

Loaded and explored the raw Titanic dataset using Pandas.

Added new column features, including a binary `is_child` feature derived from age.

Practiced vectorized operations, boolean masking, and conditional column creation without loops.

Partitioned between, processed, or raw, made the attributes non-null,

Was successfully able to broadcast, DataFrame manipulation, handling missing values.

Learning outcome: using numPy and Pandas to manipulate data, create some new columns

processed:

The screenshot shows a Jupyter Notebook environment. In the code cell, the following Python code is run:

```
import numpy as np
import pandas as pd
df = pd.read_csv("../data/processed/titanic3_eda.csv")
df1 = pd.read_csv("../data/raw/titanic3.csv")
print(df.shape)
df.head()
```

The output cell shows the execution time (0.0s) and the shape of the DataFrame (1309, 20). Below the code cell, the DataFrame is displayed as a table with the following columns:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest	money_wasted	cause_death	ticket/average	money wasted	fare_over_mean	age_minus_mean
0	1	1	Allan, Miss. Elisabeth Walton	female	29.0000	0	0	24160	211.3375	B5	S	2	NaN	St Louis, MO	0.00	14.649231	6.350078	-2		
1	1	1	Allison, Master, Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	11	NaN	Montreal, PQ / Chesterville, ON	0.00	10.504955	4.553638	-2		

At the bottom of the interface, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, XRTOS, SERIAL MONITOR, and JUPYTER. The TERMINAL tab is active, showing the command to collect numpy from xgboost.

Raw:

The screenshot shows a Visual Studio Code (VS Code) interface with several panes open, illustrating a Jupyter Notebook development workflow.

- Left Sidebar:** Shows the file tree. It includes a 'MACHINE LEARNING' folder containing 'processed' (with 'titanic3_eda.csv') and 'raw' (with 'titanic3.csv', 'titanic3.xls'). A 'notebooks' folder contains 'eda.ipynb' (marked as modified), 'machine.ipynb' (marked as untracked), 'new array.npy' (marked as new), and 'note.txt' (marked as untracked). There are also 'reports', 'src', '.gitignore', 'LICENSE', 'README.md', and 'requirements.txt' files.
- Top Center:** The main code editor pane displays the content of 'eda.ipynb'. The code imports numpy and pandas, reads CSV files, and prints the first few rows of the processed dataset.
- Bottom Center:** A preview pane shows the first 20 rows of the 'titanic3_eda.csv' dataset, which includes columns like pclass, survived, name, sex, age, sibsp, parch, ticket, fare, cabin, embarked, boat, body, and home.dest.
- Bottom Navigation:** Includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, XRTOS, SERIAL MONITOR, and JUPYTER.
- Bottom Right:** A terminal window titled 'TERMINAL' shows command-line activity related to pip and Python packages.
- Bottom Status Bar:** Shows the timeline, search bar, and various system icons.

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with the following details:

- File Tree (Left):** Shows a project structure with notebooks like `machine.ipynb`, `eda.ipynb`, and `notebooks/eda.ipynb`. Other files include `raw/titanic3.csv`, `raw/titanic3.xls`, `src`, `.gitignore`, and `LICENSE`.
- Editor Area (Top):** Displays the content of `eda.ipynb`, specifically the code to read the `titanic3.csv` file and print its first few rows.
- Terminal (Bottom):** Shows the output of a pip command to upgrade pip, indicating a new release is available.
- Right Panel:** Titled "Build with Agent", it includes a message about AI responses being inaccurate, a "Generate Agent Instructions" button, and a "Docker" notification.

The screenshot shows a Jupyter Notebook interface within VS Code. The left sidebar displays a file tree for a 'MACHINE_LEARNING' folder containing files like .venv, data, processed (titanic3_eda.csv), raw (titanic3.csv, titanic3.xls), notebooks (eda.ipynb, machine.ipynb, new_array.ipynb), reports, src, .gitignore, LICENSE, README.md, and requirements.txt. The main area has two tabs open: 'eda.ipynb' and 'Machine_Learning'. The 'Machine_Learning' tab shows a Jupyter notebook with the following code and output:

```

from sklearn.model_selection import train_test_split
df = pd.read_csv('titanic3.csv')
df.drop(['name', 'ticket', 'cabin'], axis=1, inplace=True)
df['age'].fillna(df['age'].mean(), inplace=True)
df['embarked'].fillna('S', inplace=True)
df['sex'] = df['sex'].map({'male': 0, 'female': 1})
df['survived'] = df['survived'].map({0: 'No', 1: 'Yes'})
df['pclass'] = df['pclass'].map({1: '1st', 2: '2nd', 3: '3rd'})
df['sibsp'] = df['sibsp'].map({0: 'Alone', 1: 'With Sibling', 2: 'With Spouse', 3: 'With Family'})
df['parch'] = df['parch'].map({0: 'Alone', 1: 'With Sibling', 2: 'With Spouse', 3: 'With Family'})
df['home.dest'] = df['home.dest'].map({'South': 'South', 'North': 'North', 'Other': 'Other', 'Unknown': 'Unknown'})
df['embarked'] = df['embarked'].map({'S': 'South', 'C': 'Central', 'Q': 'Quay'})
df['sex'] = df['sex'].map({0: 'Male', 1: 'Female'})
df['survived'] = df['survived'].map({0: 'No', 1: 'Yes'})
df['pclass'] = df['pclass'].map({1: '1st', 2: '2nd', 3: '3rd'})
df['sibsp'] = df['sibsp'].map({0: 'Alone', 1: 'With Sibling', 2: 'With Spouse', 3: 'With Family'})
df['parch'] = df['parch'].map({0: 'Alone', 1: 'With Sibling', 2: 'With Spouse', 3: 'With Family'})
df['home.dest'] = df['home.dest'].map({'South': 'South', 'North': 'North', 'Other': 'Other', 'Unknown': 'Unknown'})
df['embarked'] = df['embarked'].map({'S': 'South', 'C': 'Central', 'Q': 'Quay'})

# Data Cleaning
df.dropna(inplace=True)
df['age'].fillna(df['age'].mean(), inplace=True)
df['embarked'].fillna('S', inplace=True)

# Feature Engineering
df['family_size'] = df['sibsp'] + df['parch'] + 1
df['is_alone'] = df['family_size'].apply(lambda x: 1 if x == 1 else 0)
df['is_male'] = df['sex'].apply(lambda x: 1 if x == 'Male' else 0)
df['is_female'] = df['sex'].apply(lambda x: 1 if x == 'Female' else 0)
df['is_south'] = df['home.dest'].apply(lambda x: 1 if x == 'South' else 0)
df['is_north'] = df['home.dest'].apply(lambda x: 1 if x == 'North' else 0)
df['is_other'] = df['home.dest'].apply(lambda x: 1 if x == 'Other' else 0)
df['is_unknown'] = df['home.dest'].apply(lambda x: 1 if x == 'Unknown' else 0)
df['is_c'] = df['pclass'].apply(lambda x: 1 if x == '3rd' else 0)
df['is_q'] = df['pclass'].apply(lambda x: 1 if x == '2nd' else 0)
df['is_s'] = df['pclass'].apply(lambda x: 1 if x == '1st' else 0)
df['is_sibsp'] = df['sibsp'].apply(lambda x: 1 if x > 0 else 0)
df['is_parch'] = df['parch'].apply(lambda x: 1 if x > 0 else 0)

# Model Selection
X = df[['age', 'family_size', 'is_alone', 'is_male', 'is_female', 'is_south', 'is_north', 'is_other', 'is_unknown', 'is_c', 'is_q', 'is_s', 'is_sibsp', 'is_parch']]
y = df['survived']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

```

The right sidebar includes a 'Build with Agent' feature and a terminal window showing pip upgrade logs.

Days 3–4: (Scikit-Learn)

Did study more on machine learning, about logistic regression and linear regression

The screenshot shows a Jupyter Notebook interface within VS Code. The left sidebar displays a file tree for a 'MACHINE_LEARNING' folder containing files like .venv, data, processed (titanic3_eda.csv), raw (titanic3.csv, titanic3.xls), notebooks (eda.ipynb, machine.ipynb, new_array.ipynb), reports, src, .gitignore, LICENSE, README.md, and requirements.txt. The main area has three tabs open: 'eda.ipynb', 'Machine_Learning', and 'note.txt'. The 'note.txt' tab shows a Jupyter notebook with the following text:

```

Linear Regression
z=w1x1+w2x2...+b
we train, on different sets of weights, and bs, until we get least error, that is linear Regression. We get values from -infinity to infinity
example: price = 50.000 * bedrooms + 20.000 * bathrooms + 10.000
Logistic Regression
basically same as linear Regression, but uses a sigmoid function
p=0<z>/1/(1+exp(-z))
p=Py(x)
log(1-p)/p=w1x1...+b
utilizes log loss
Overfitting
THIS exact dog with this exact spot pattern is a dog
Underfitting
ALL animals are dogs
Bias-Variance Tradeoff
Bias = how wrong the model is by design, High bias: Underfitting
Variance = how sensitive the model is to data, High variance: Overfitting
Managing Bias and Variance, is what our intent is for our purpose.
Regularization
Training minimizes loss, regulation adds penalty, (basically imagine a knob, changes weight by how much each time)
Loss + λ * Penalty(weights)
L2 Regularization(Ridge)
Penalty: ∑w_i^2
L1 Regularization(lasso)
Penalty: ∑|w_i|

```

The right sidebar includes a 'Build with Agent' feature and a terminal window showing Docker installation recommendations.

Learned about regulation, L1 and L2, variance, overfitting and underfitting,

I also learned about Bias-Variance Tradeoff, I also learned about generalization and why we must keep some data for testing separate from training data.

Days 5–6: XGBoost

Implemented an **XGBoost classifier** using a Scikit-Learn Pipeline

Trained an XGBoost model on the Titanic dataset.

Plotted the ROC curve and verified strong separation from random guessing

The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows the project structure with files like `titanic3.csv`, `titanic3.xls`, and `eda.ipynb`.
- TERMINAL:** Displays Python code for training an XGBoost classifier using a Pipeline. The code includes imports for `XGBClassifier` and `ColumnTransformer`, and defines a `xgb_pipe` Pipeline with a `ColumnTransformer` containing `SimpleImputer` for numerical features and `OneHotEncoder` for categorical features, followed by an `XGBClassifier`. The terminal output shows the pipeline being fitted to the data.
- CODEVIEW:** Shows the Python code for the XGBoost pipeline.
- SCHEMATIC:** A diagram of the Pipeline structure, showing the flow from input data through the `ColumnTransformer` (with `num` and `cat` components) to the final `XGBClassifier`.
- CHAT:** A sidebar with a "Build with Agent" feature.

The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows the project structure with files like `titanic3.csv`, `titanic3.xls`, and `eda.ipynb`.
- TERMINAL:** Displays Python code for plotting an ROC curve. The code uses `roc_curve` and `plot_roc_curve` from scikit-learn. The terminal output shows the ROC curve being plotted.
- CODEVIEW:** Shows the Python code for plotting the ROC curve.
- SCHEMATIC:** A plot of the True Positive Rate versus False Positive Rate, titled "XGBoost". The curve starts at (0,0) and rises sharply, then levels off towards 1.0, indicating good model performance. A dashed diagonal line represents a random classifier.
- CHAT:** A sidebar with a "Build with Agent" feature.

Day 7: Evaluation

Compared baseline models and XGBoost using probabilistic outputs

Learned why accuracy is misleading for imbalanced datasets.

Learned why accuracy is misleading for imbalanced datasets.

ROC-AUC measures ranking ability, not just correctness.

