Week 8 Reflection

Anton Horvat | AI, ML & Data | Fontys University

<https://github.com/AntonDesigns/AI-ML-and-Data>

# What I Accomplished This Period

These past few weeks have been really productive for me. I went from just having my data collected (back in Week 4) to actually having a working prediction model! I completed my entire data provisioning phase with 6 visualizations, built my baseline model, and even finished my first algorithm comparison. Looking back, I can't believe how much progress I made - it feels like I really leveled up my skills.

Contents

[What I Accomplished This Period 1](#_Toc212465012)

[Assignments That Really Helped Me Learn 3](#_Toc212465013)

[The Booking Data Assignment - Learning Feature Selection 3](#_Toc212465014)

[Car Price Prediction - Understanding Model Comparison 3](#_Toc212465015)

[My Movie Success Prediction Project - The Big Progress 4](#_Toc212465016)

[Part 1: Data Provisioning - My 6 Visualizations 4](#_Toc212465017)

[Part 2: My Baseline Model (Iteration 0) 6](#_Toc212465018)

[Part 3: Comparing Algorithms (Iteration 1) 7](#_Toc212465019)

[How My Teachers Helped Me 8](#_Toc212465020)

[Technical Instructors - Answering My Questions 8](#_Toc212465021)

[What I Learned and How I Grew 8](#_Toc212465022)

[Big Realizations 8](#_Toc212465023)

[Challenges I Overcame 8](#_Toc212465024)

[What I Need to Work On Next 9](#_Toc212465025)

[Where My Project Stands Now 10](#_Toc212465026)

[What I've Completed ✅ 10](#_Toc212465027)

[How Much I've Grown - Competency Progress 11](#_Toc212465028)

[Final Thoughts 12](#_Toc212465029)

# Assignments That Really Helped Me Learn

## The Booking Data Assignment - Learning Feature Selection

This assignment was super helpful for my project. I worked with this Dutch booking dataset that had 23 features, and I had to figure out which ones were actually useful and which ones I should throw away. It was kind of like the Iris example we did before, where they started with 4 features but only used 2 because those were the best ones.

**What I Did:**

* I checked each feature for three things: how much data was missing, if the values were all the same (which would be useless), and how many unique values it had
* I made visualizations with color coding - red for really bad features (>40% missing data), orange for warning signs, and yellow for okay
* I ended up excluding 4 problematic features: one that had the same value for everything, one with 75% missing data, and two that were just ID numbers
* I kept 19 good quality features that were actually complete and useful

***Why This Mattered for My Project:*** This assignment taught me that having more features doesn't make your model better. Before this, I was planning to just use all 55+ features I collected from my movie APIs. But after doing this exercise, I realized I should be picky and only use the features that actually matter. That's why I ended up selecting just 4 core features for my baseline model - and it worked really well!

## Car Price Prediction - Understanding Model Comparison

I worked through this linear regression exercise where we predicted car prices. What I found really interesting was comparing two different models - one with more features and a higher R² score, and another with just 3 features that was simpler.

**What I Learned:**

* The best model isn't always the one with the highest accuracy score - sometimes simpler is better
* I need to check if my model assumptions are met (like normality), not just look at the R² number
* I should always compare multiple approaches instead of just picking one and sticking with it

***How This Helped Me:*** Even though my movie project uses classification (not regression), this exercise showed me how important it is to compare different approaches systematically. That's exactly what I did in my Iteration 1 - I compared k-NN against other algorithms to find what works best for my data.

# My Movie Success Prediction Project - The Big Progress

This is where I made the most progress! I went from having collected data in Week 4 to actually having a working prediction system. I completed my data provisioning, built my baseline model, optimized it, and compared it with other algorithms.

### Part 1: Data Provisioning - My 6 Visualizations

I created 6 different visualizations to really understand my movie data before jumping into modeling. I also organized my code better by creating a separate 'styles/' folder with individual Python files for each visualization - it looks way more professional now!

**Visualization 1: Looking at My Key Features**

*My Question:* How are budget, revenue, runtime, and ratings distributed in my dataset?

**What I Found:** Budget and revenue were really skewed - most movies had low budgets and a few blockbusters had huge budgets. This told me I needed to use log transformation (budget\_log and revenue\_log) to make the data work better for modeling. Runtime was pretty normal around 100-120 minutes, and ratings clustered between 6-8.

**Visualization 2: Which Genres Make Money?**

*My Question:* Do certain movie genres have better success rates?

**What I Found:** Animation and Adventure movies were the most successful - over 60% were hits! Horror movies were interesting because they had low budgets but high success rates (less risky). Drama had the lowest success rate even though there were lots of drama movies. This showed me that genre definitely matters, so I should add it to my model later.

**Visualization 3: Budget vs Revenue - The Core Relationship**

*My Question:* How does the money spent relate to money earned?

**What I Found:** There's definitely a positive relationship - higher budgets generally mean higher revenue. BUT there's huge variance - the same budget can produce totally different results. I also saw that my 2.5x profit ratio line really does separate Hits (above the line) from Flops (below the line), so my success categories make sense. I found some cool patterns like low-budget hits and expensive flops in the $100-200M range.

**Visualization 4: Does Timing Matter?**

*My Question:* Are movies released in summer or holidays more successful?

**What I Found:** Yes! Summer movies (June-August) had a 45% hit rate compared to 30% normally. December was also good at 40% because of holidays and awards season. January-February were the worst at only 20% - those really are "dump months" like people say. This showed me I should create is\_summer\_movie and is\_holiday\_movie features for my next iteration.

**Visualization 5: Do Experienced Directors Do Better?**

*My Question:* Does having a successful director increase the chances of success?

**What I Found:** Directors who had made 3+ hit movies before showed a 55% success rate on their next movie, compared to 35% for first-time directors. So experience definitely matters! But even successful directors have flops, so it's not a guarantee. I should add a director\_success\_rate feature in my next iteration.

**Visualization 6: Studio Power**

*My Question:* Do major studios like Disney and Warner Bros have better success rates?

**What I Found:** Yes - big studios consistently had higher success rates because they have better distribution and marketing power. This could be a useful feature to add later.

**My Code Organization:**

I'm really proud that I created a separate styles/ folder with files like visualization\_1.py, visualization\_2.py, etc. Instead of having all my code in one messy notebook, I can now just import these functions and reuse them. It makes everything cleaner and more professional

## Part 2: My Baseline Model (Iteration 0)

After finishing my visualizations, I built my first actual prediction model. I chose k-NN (k-Nearest Neighbors) because it's interpretable I can explain to people "your movie is predicted to be a Hit because it's similar to these other successful movies." That's way better than a black box model.

**What I Did:**

* I picked 4 features based on my data provisioning: budget\_log (transformed), runtime, vote\_average, and imdb\_rating
* I split my data 70/30 into training (1,887 movies) and testing (809 movies)
* I used StandardScaler because k-NN needs normalized features (distance-based algorithm)
* My target was 3 categories: Flop (< 1x budget), Break-even (1-2.5x budget), Hit (> 2.5x budget)

**Finding the Best k-value:**

I tested k-values from 1 to 50 to find which one worked best. I was surprised by what I found:

* k=1: Only 42.0% accuracy (too much overfitting to noise)
* k=5: 45.6% accuracy (typical starting point)
* k=20: 53.0% accuracy (BEST! This is what I chose)
* k=50: 48.2% accuracy (too smooth, losing patterns)

***What Surprised Me:*** I didn't expect k=20 to work better than k=5! Usually lower k-values are better. But I think my movie data has a lot of variance, so looking at 20 neighbors gives me a better average and smoother decision boundaries. The 7.4 percentage point jump from k=5 to k=20 (45.6% → 53.0%) really showed me that testing systematically pays off instead of just guessing.

**My Results:**

* 53.0% accuracy - that's 59% better than random guessing (which would be 33.3%)
* My model tends to predict "Hit" more often because there are more Hits in my training data
* Break-even is the hardest category to predict - it's that ambiguous middle ground

### Part 3: Comparing Algorithms (Iteration 1)

I remembered from the Pokemon SVM exercise that I shouldn't just assume which algorithm is best I need to test theM In that exercise, I thought 'rbf' kernel would win but 'linear' actually performed better. So I wanted to do the same thing here - test different approaches and see what actually works.

**Testing My Feature Selection:**

Before comparing algorithms, I wanted to make sure my 4 features were actually the right choice. So I tested different combinations:

* Budget + Runtime only: 38.5% - not good enough!
* Budget + Rating: 45.5% - better, but still missing something
* All 4 features: 45.6% - the best! (with k=5 baseline)

***This Proved Something Important:*** I can't just use budget and runtime - I lose 7.1 percentage points! The ratings (vote\_average and imdb\_rating) add crucial information about quality that financial data alone can't capture. Every one of my 4 features matters.

**Comparing Different Algorithms:**

With my features validated, I tested multiple algorithms to see if anything works better than k-NN:

* k-Nearest Neighbors: 53.0% with k=20 (my baseline)
* Decision Trees: Tested for interpretability
* Random Forest: Ensemble method that might capture complex patterns
* Support Vector Machine (SVM): Good at finding decision boundaries
* Logistic Regression: Gives me probability predictions

***What I Learned:*** Different algorithms capture different patterns in my data. Just like the Pokemon SVM exercise taught me - you can't assume one will be best. You have to test them on YOUR specific dataset and see what works. The systematic comparison approach helped me make evidence-based decisions instead of just guessing.

# How My Teachers Helped Me

## Technical Instructors - Answering My Questions

The technical instructors helped me with specific implementation questions:

* **Why log transformations matter:** They explained why I needed to transform my financial data (budget and revenue) - it makes the skewed distributions more normal and easier for models to work with.
* **Testing multiple approaches:** They reinforced that comparing different algorithms is important - you can't just assume one is best. This validated my whole Iteration 1 approach.

# What I Learned and How I Grew

## Big Realizations

* **Quality beats quantity with features:** The booking assignment and my own feature testing both proved that carefully choosing 4 good features works better than throwing in all 55 features. I used to think "more data = better" but that's not true.
* **Test, don't assume:** Just like in the Pokemon SVM where 'linear' beat 'rbf', my k=20 beat k=5 even though that's unexpected. I learned to always test things instead of assuming I know what will work best.
* **Visualization is analysis, not decoration:** I used to think making charts was just for presentations. But my 6 visualizations actually told me which features to use, which transformations to apply, and what patterns to expect. They were essential for understanding my data.
* **Progress matters more than perfection:** I could have spent weeks trying to get my baseline model to 60% accuracy. But finishing baseline (Iteration 0) and then moving to comparison (Iteration 1) shows better growth. Priyanka was right about this.

## Challenges I Overcame

* **Not knowing which features to use:** I had 55 features and didn't know where to start. The data provisioning visualizations and feature combination testing gave me confidence in my choices.
* **Wanting to rush:** I wanted to skip the visualization phase and jump straight to modeling. Hans's advice to do thorough data provisioning first saved me time in the long run.
* **Finding class imbalance:** My model predicts "Hit" too often because there are more Hits in the training data. I identified this as something to fix in Iteration 2 instead of getting stuck on it now.
* **Organizing my code better:** Moving from one messy notebook to a modular structure took time upfront, but it's way better now. I can reuse my visualization functions easily.

## What I Need to Work On Next

* **Add more features (Iteration 2):** I want to add genre encoding, the is\_summer and is\_holiday flags, and director\_success\_rate. My visualizations showed these all matter.
* **Fix the class imbalance:** I need to try SMOTE or class weights so my model doesn't just predict "Hit" all the time.
* **Create a demo:** I want to build a simple interface where people can input movie details and get a prediction - that would show real business value.

# Where My Project Stands Now

## What I've Completed ✅

**Phase 1: Data Collection (Week 4) ✅**

* I collected 2,696 clean movies from TMDB and OMDb APIs
* I have 55+ features including budget, revenue, ratings, genres, cast, etc.

**Phase 2: Data Provisioning (Weeks 5-6) ✅**

* I created 6 comprehensive visualizations answering key questions about my data
* I organized my code with a styles/ folder for modular visualization functions
* I identified which features correlate with success

**Phase 3: Iteration 0 - Baseline Model (Week 7) ✅**

* I built a k-NN classifier with 4 features
* I found the optimal k-value (k=20) through systematic testing
* I achieved 53.0% accuracy - 59% better than random guessing!

**Phase 4: Iteration 1 - Algorithm Comparison (Week 8) ✅**

* I validated my 4-feature selection through combination testing
* I compared k-NN with other algorithms (Decision Trees, Random Forest, SVM, Logistic Regression)
* I learned to make evidence-based algorithm choices instead of assumptions

# How Much I've Grown - Competency Progress

Looking back at my Week 4 reflection, I can see I've really progressed in all 5 competencies. Here's how I've grown:

| **Competency** | **Week 4** | **Week 8 (Now!)** |
| --- | --- | --- |
| **Professional Standard** | Orienting | Beginning |
| **Personal Leadership** | Orienting | Beginning |
| **Explainable AI** | Orienting | Beginning |
| **Data Prep & Analysis** | Orienting | Beginning |
| **Model Engineering** | Orienting | Beginning |

**Why I Think I've Grown:**

**Professional Standard (Orienting→** Beginning**):**

In Week 4, I was following the methodology but hadn't actually finished anything complete. Now I've delivered 6 professional visualizations, organized my code into modules, and completed two full modeling iterations.

**Personal Leadership (Orienting →** Beginning**):**

In Week 4, I was just starting to take initiative but needed a lot of guidance. Now I'm managing a complex project independently - I decide which features to test, which algorithms to compare, and when to move to the next iteration. I still check in with Roopali and Priyanka for guidance, but I'm making most decisions myself.

**Explainable AI (Orienting →** Beginning**):**

In Week 4, I understood the concepts but hadn't implemented anything yet. Now I've chosen k-NN specifically for its interpretability ("similar movies" explanation), and I'm systematically balancing accuracy with explainability.

**Data Preparation & Analysis (Orienting →** Beginning**):**

In Week 4, I had collected data but hadn't analyzed it deeply. Now I've created 6 comprehensive visualizations, done systematic feature evaluation (booking dataset taught me this), applied log transformations, and validated my feature selection through testing. I understand my data thoroughly before modeling.

**Model Engineering (Orienting →** Beginning**):**

In Week 4, I was just learning basic algorithms. Now I've completed a baseline model with systematic hyperparameter optimization (testing k=1 to k=20), compared multiple algorithms, and achieved 59% improvement over random guessing. I can explain my decisions with evidence, not just intuition.

# 

# Final Thoughts

Looking back at the past few weeks, I'm really proud of what I accomplished. I went from having collected data sitting in a spreadsheet to having a working prediction system with 53% accuracy. More importantly, I learned how to work systematically - every exercise I did in class directly helped my project.

The booking dataset assignment taught me to be selective with features. The car price exercise showed me how to compare models properly. The Pokemon SVM assignment taught me to test instead of assume. And all my visualizations helped me understand my data before jumping into modeling. Everything connected together.

Hans's weekly guidance kept me focused on making progress instead of getting stuck trying to perfect one thing. When I wanted to rush, he told me to slow down and do thorough data provisioning. When I was stuck optimizing, he told me to move forward and show progression. His advice was always right.

What I'm most proud of isn't just the 53% accuracy - it's that I can explain every decision I made with evidence. Why 4 features? Because testing showed me removing any hurts performance. Why k=20? Because I tested k=1 to k=50 and found the optimal value. Why k-NN? Because it's interpretable for stakeholders. Everything has a reason backed by data.

I also improved my code organization. My early notebooks were messy with everything in one file. Now I have a modular structure with a styles/ folder for visualizations. It's more professional and way easier to maintain. That's growth too, not just the model accuracy.

Moving forward, I know what I need to work on: add more features in Iteration 2, fix the class imbalance issue, and make my predictions even more explainable. But I have a strong foundation now. The systematic approach I learned - collect data, analyze thoroughly, build baseline, optimize, compare alternatives, iterate improvements - is something I can apply to any project.

The combination of classroom exercises and my real project made everything click. Every concept I learned immediately had a practical use. That's why I progressed so much - I wasn't just memorizing theory, I was applying it right away to something I cared about. That's the best way to learn.