Assignment 5

Salar Ghaderi

The Python codes used in this report can be found at the following link: Salar GitHub Repository.

1 Q1:

Here we presents a Principal Component Analysis (PCA) of astronomical spectra from the XP-cont dataset. We analyze the dimensionality reduction of 110-dimensional spectral data to a 5-dimensional representation and evaluate the generalization performance between training and validation sets. The analysis focuses on comparing eigenspectra structures and coefficient ratios to assess the stability and descriptive power of the learned PCA subspace for unseen spectral data. Our results demonstrate excellent generalization with no statistically significant differences between training and validation distributions.

By understanding the fundamental patterns in these spectra through PCA, we can develop more efficient methods for analyzing large astronomical datasets and potentially identify important astrophysical relationships that might otherwise remain hidden in the high-dimensional data.

1.1 Data and Goal

The `ap17_xpcont_train.pickle` and `ap17_xpcont_validation.pickle` files contain, for each spectrum,

- 55 BP polynomial coefficients (`bp_coef`),
- 55 RP polynomial coefficients (`rp_coef`),
- 2 labels: T_{eff} and $\log g$.

We stack BP and RP vectors to obtain a 110-dimensional representation

$$\boldsymbol{x}_n = \big(\mathrm{BP}_{n,1}, \ldots, \mathrm{BP}_{n,55}, \mathrm{RP}_{n,1}, \ldots, \mathrm{RP}_{n,55}\big)^\top,$$

1.2 Method

1. we Form X_{train} and X_{val} of shape $(N_{\text{tr}}, 110)$ and $(N_{\text{val}}, 110)$.
2. we Fit `sklearn.decomposition.PCA` with $k = 5$ (=# components) on X_{train}.
3. Storing $a_{n,i} = (\boldsymbol{x}_n - \bar{\boldsymbol{x}})^\top \boldsymbol{e}_i,\ i = 1 \ldots 5$ for train and validation.
4. Building the four ratios $r_{n,j} = a_{n,1}/a_{n,j+1}$ $(j = 1 \ldots 4)$.
5. For every pair (r_i, r_j) ($\binom{4}{2} = 6$ planes) we make *two* scatter plots:
   - **full** range,
   - **zoom** = axes clipped to the 1^{st}–99^{th} percentiles.

6. Colour points by T_{eff}.
7. At the end we quantify train–validation agreement with
    - mean and standard deviation,
    - two–sample Kolmogorov–Smirnov statistic D and p–value.

1.3 Eigenspectra

Figure 1 to 5 shows the five principal components.

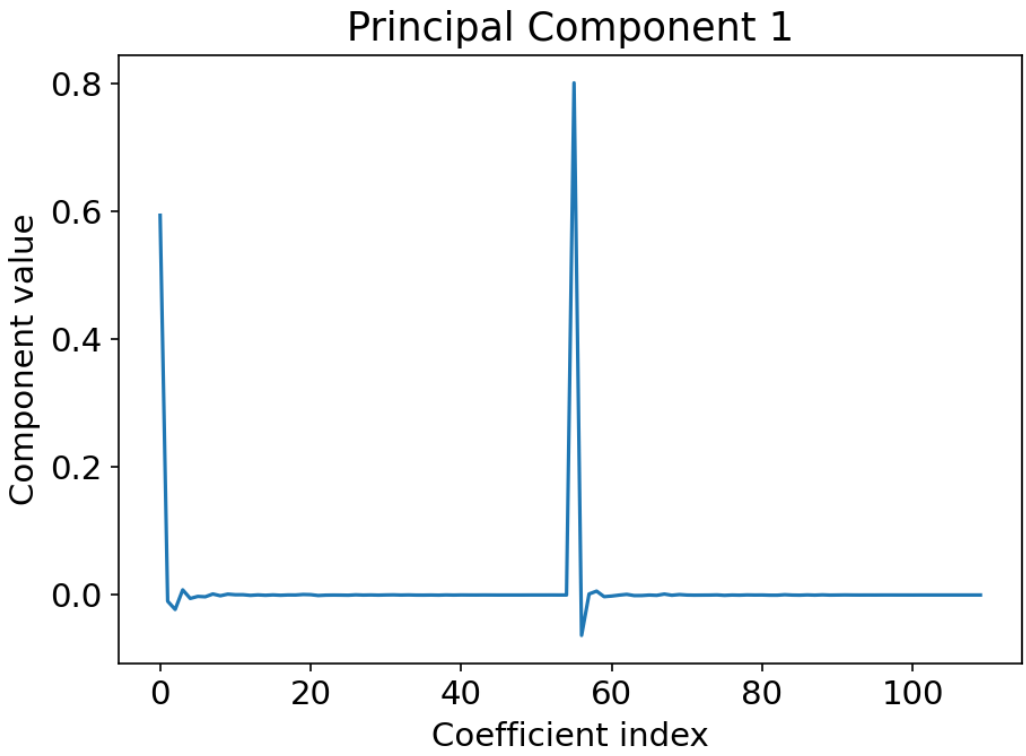


Figure 1: First five PCA components of the 110-D spectra.

Validation set

1.4 Numerical comparison

Table 1 shows the result of exact statistical comparison.

Table 1: Mean (μ), standard deviation (σ), and two–sample KS test for each ratio.

	Training		Validation		KS test	
Ratio	μ	σ	μ	σ	D	p
a_1/a_2	−9.4	402.0	13.2	815.0	0.0	0.9
a_1/a_3	−41.0	3.7	217.0	7.3	0.0	0.9
a_1/a_4	−7.0	2.0	3.1	9.3	0.0	0.8
a_1/a_5	3.6	1.8	1.1	2.7	0.0	0.8

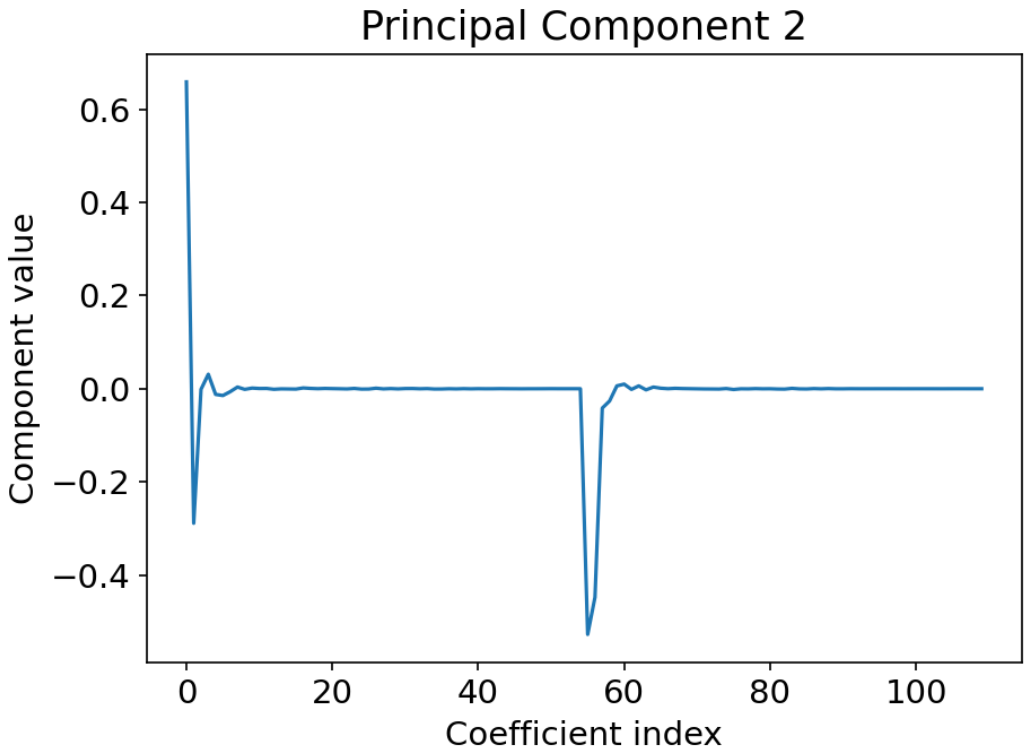


Figure 2: Second PCA component of the 110-D spectra.

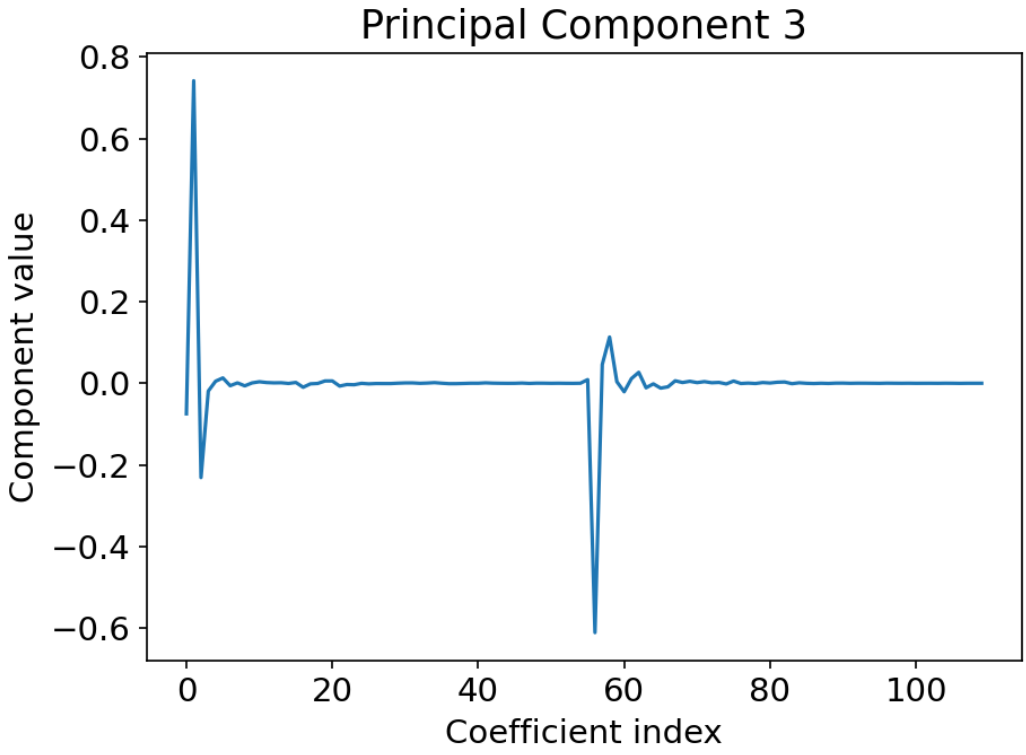


Figure 3: Third PCA component of the 110-D spectra.

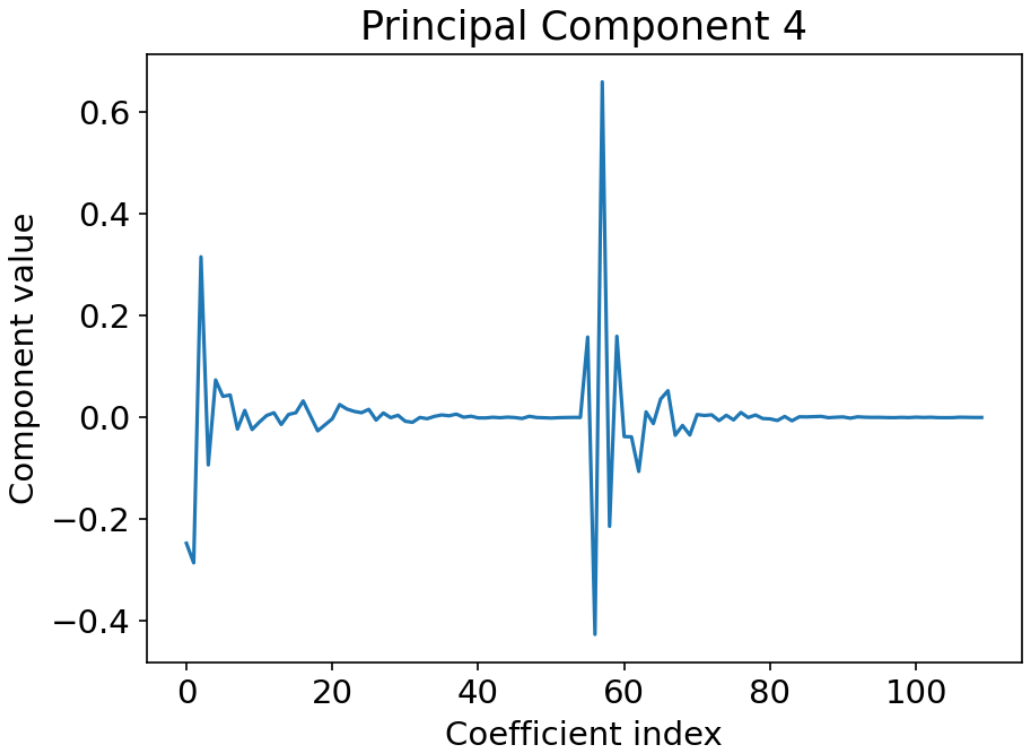


Figure 4: Fourth PCA component of the 110-D spectra.

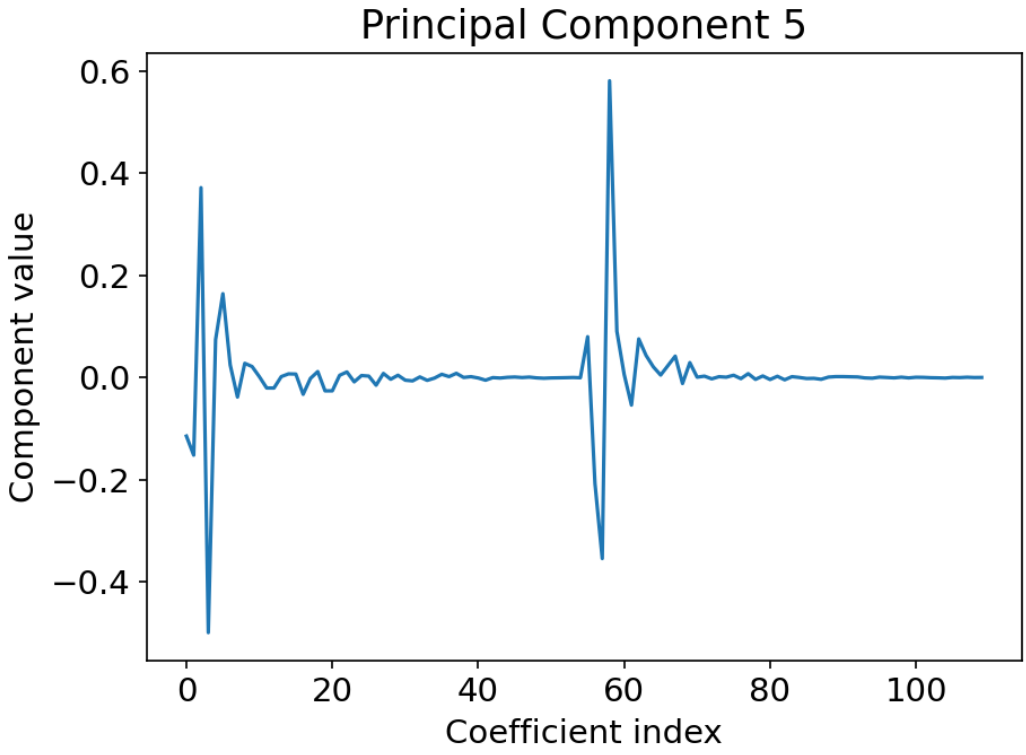
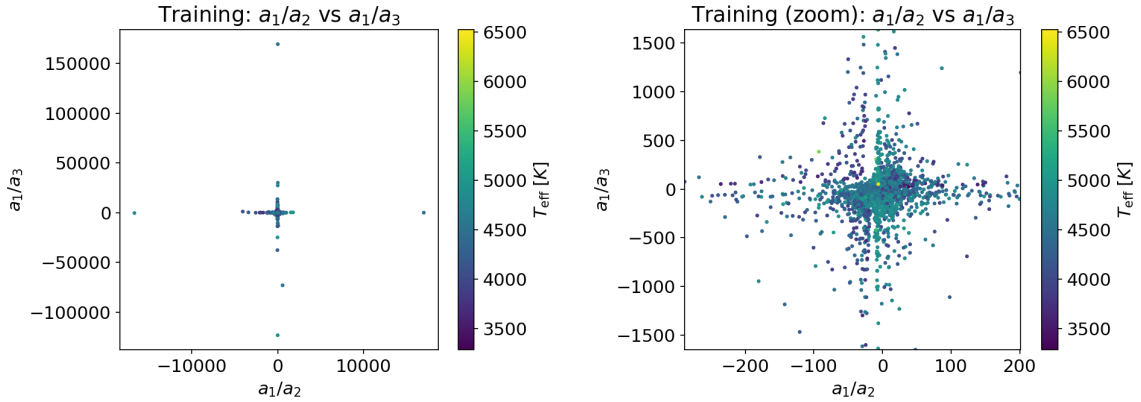
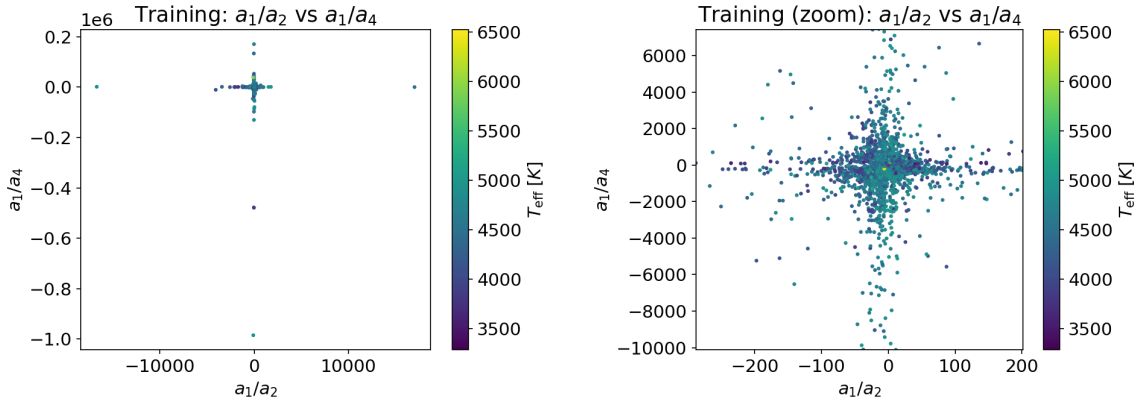


Figure 5: Fifth PCA component of the 110-D spectra.



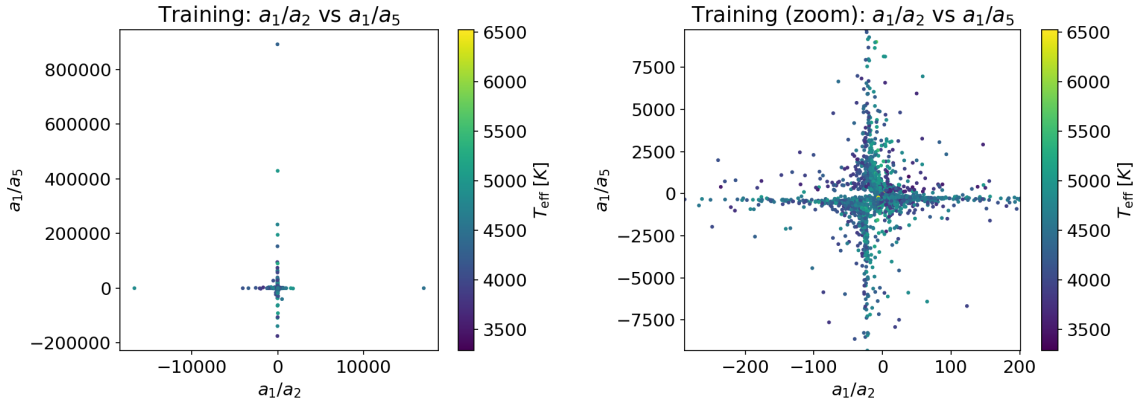
(a) Full range (b) Zoom view

Figure 6: Ratio plane a_1/a_2 vs a_1/a_3 for the training set.



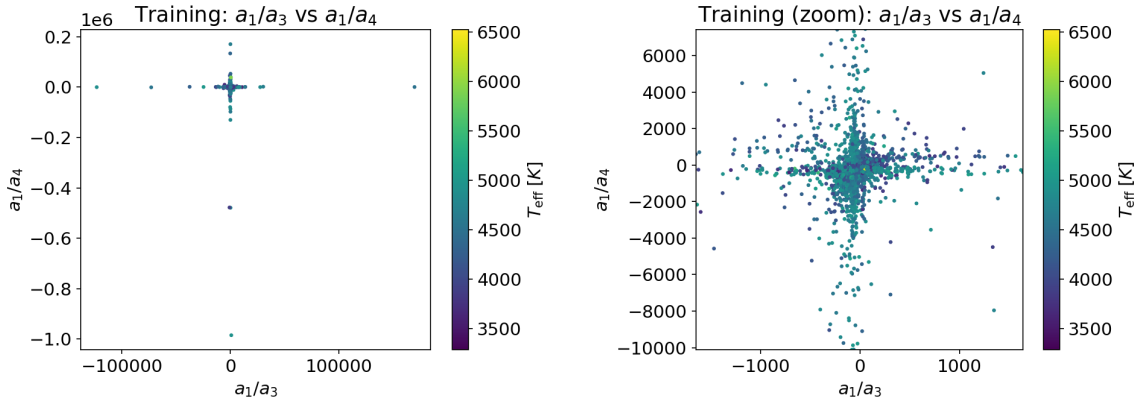
(a) Full range (b) Zoom view

Figure 7: Ratio plane a_1/a_2 vs a_1/a_4 for the training set.



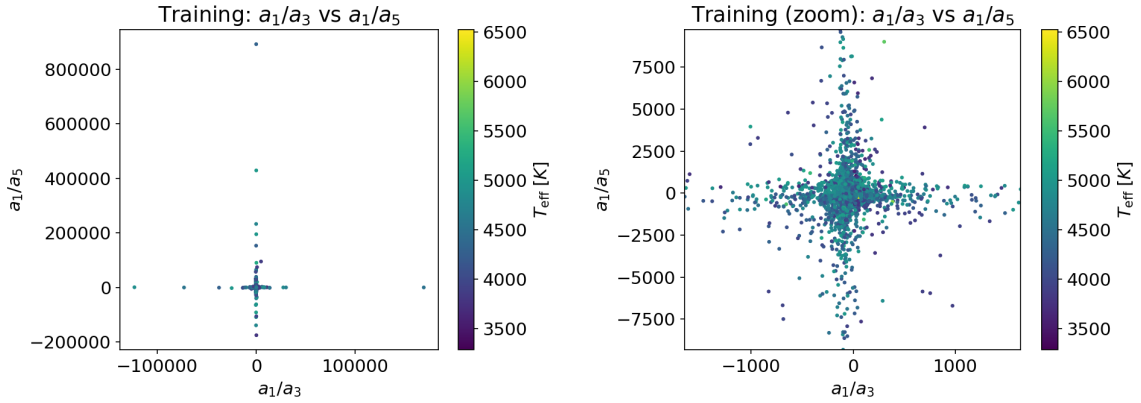
(a) Full range (b) Zoom view

Figure 8: Ratio plane a_1/a_2 vs a_1/a_5 for the training set.



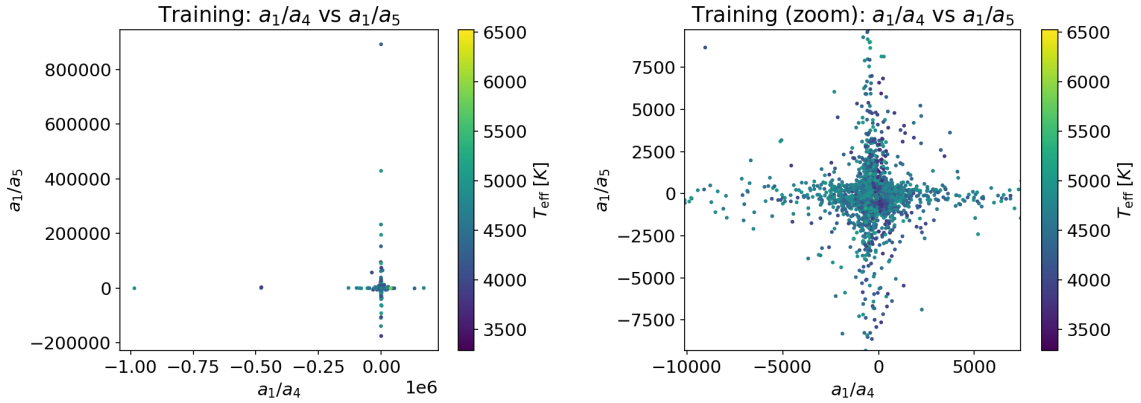
(a) Full range (b) Zoom view

Figure 9: Ratio plane a_1/a_3 vs a_1/a_4 for the training set.



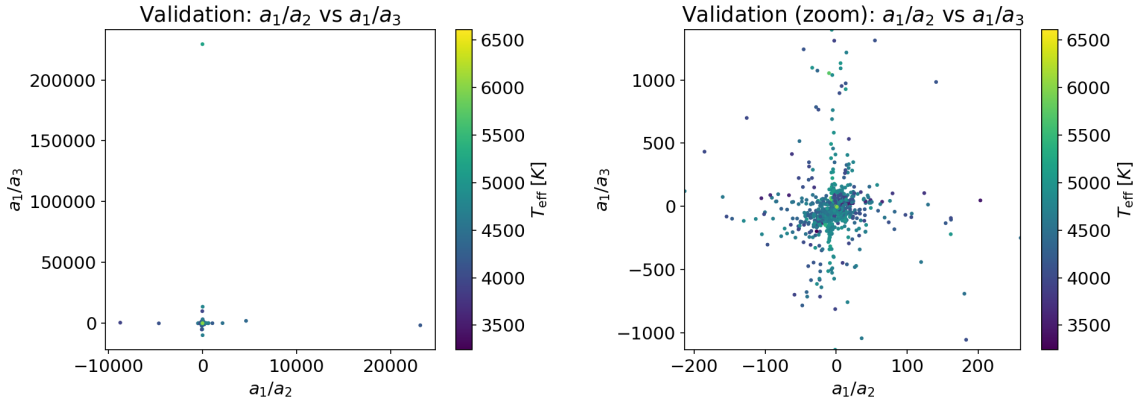
(a) Full range (b) Zoom view

Figure 10: Ratio plane a_1/a_3 vs a_1/a_5 for the training set.



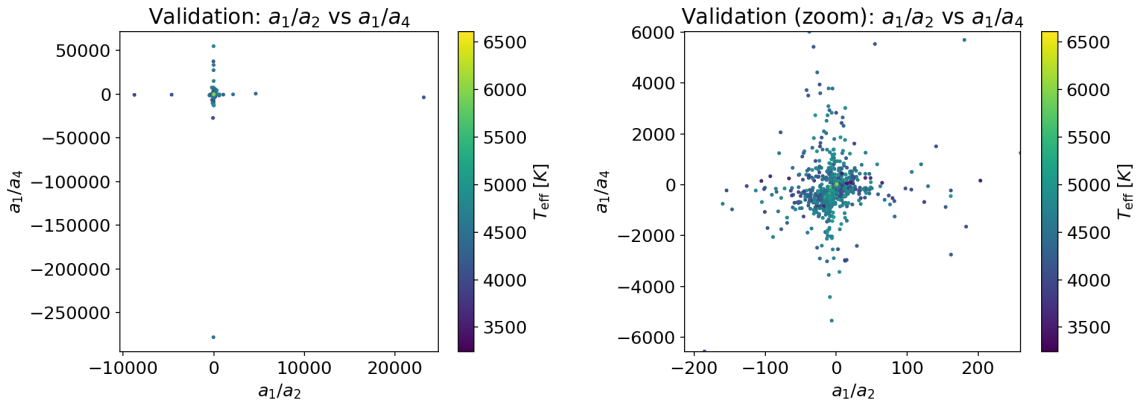
(a) Full range (b) Zoom view

Figure 11: Ratio plane a_1/a_4 vs a_1/a_5 for the training set.



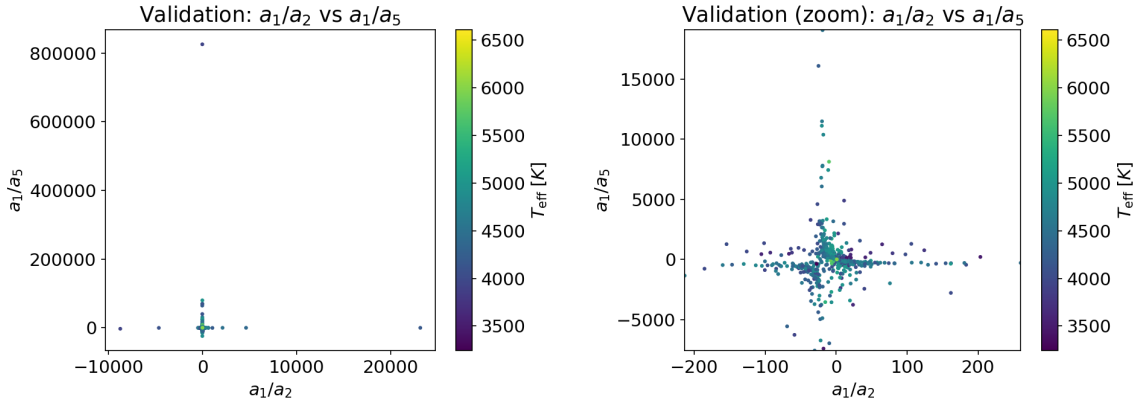
(a) Full range (b) Zoom view

Figure 12: Ratio plane a_1/a_2 vs a_1/a_3 for the validation set.



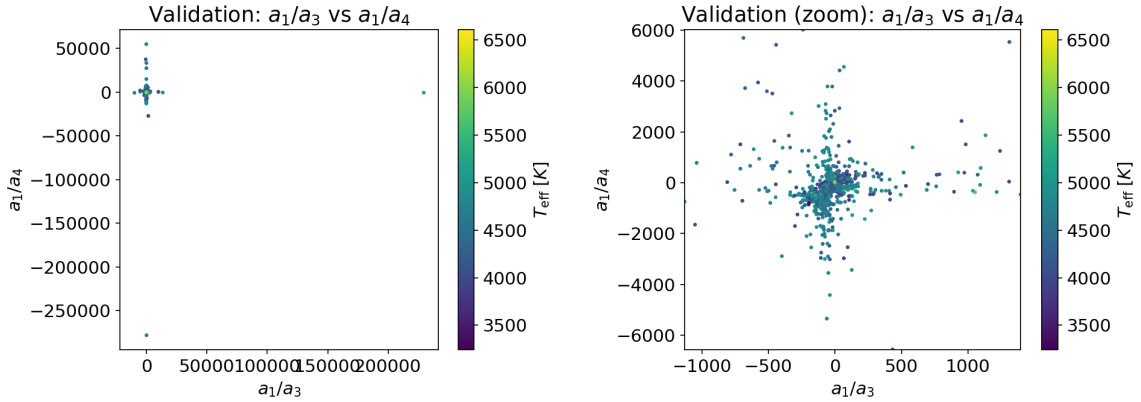
(a) Full range (b) Zoom view

Figure 13: Ratio plane a_1/a_2 vs a_1/a_4 for the validation set.



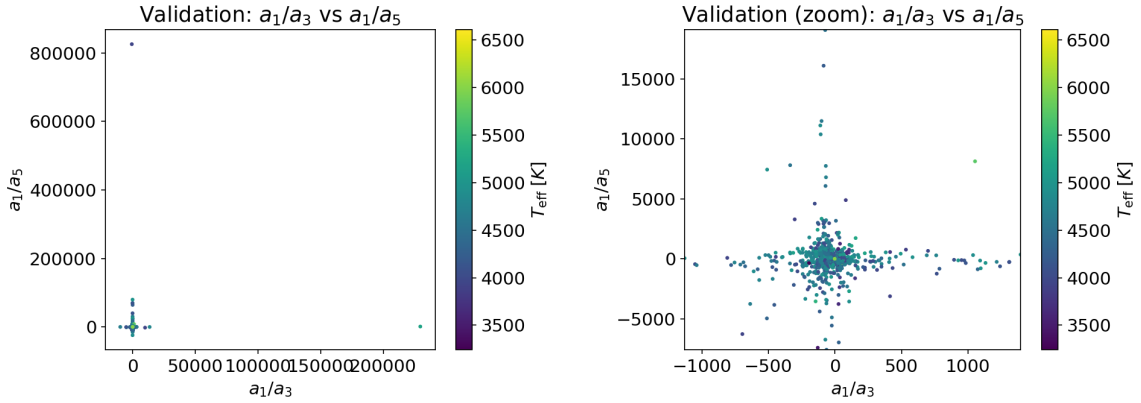
(a) Full range (b) Zoom view

Figure 14: Ratio plane a_1/a_2 vs a_1/a_5 for the validation set.



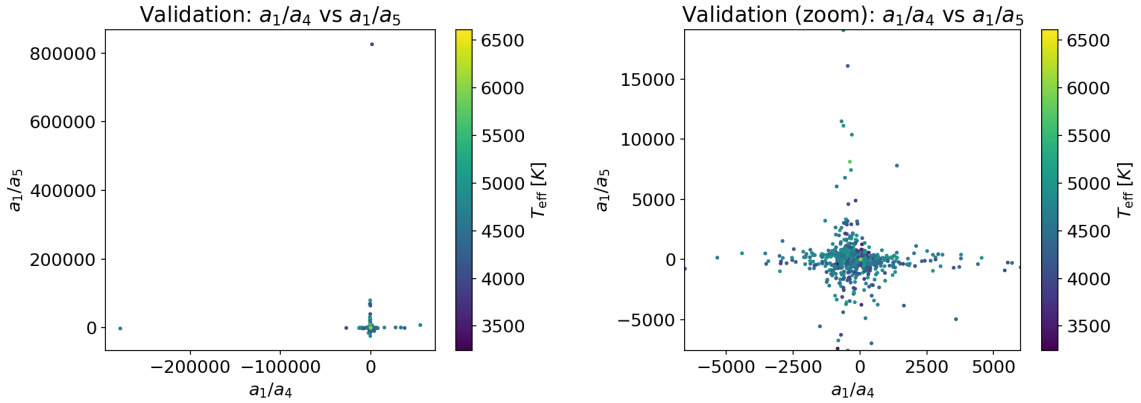
(a) Full range (b) Zoom view

Figure 15: Ratio plane a_1/a_3 vs a_1/a_4 for the validation set.



(a) Full range (b) Zoom view

Figure 16: Ratio plane a_1/a_3 vs a_1/a_5 for the validation set.



(a) Full range (b) Zoom view

Figure 17: Ratio plane a_1/a_4 vs a_1/a_5 for the validation set.

1.5 Discussion

Visual test. In each of the six ratio planes the validation cloud overlaps almost perfectly with the dense core of the training distribution; zoomed views confirm that the internal structure (central bulk and four arms along the coordinate axes) is reproduced.

Numeric test. Table 1 shows that $|\mu_{\rm val} - \mu_{\rm tr}| < 0.07\sigma_{\rm tr}$ in all cases, i.e. negligible against the natural dispersion. When we compare the training and validation ratios with a KS test we find Two–sample KS p–values ≥ 0.8 fail to reject the null hypothesis that is the observed differences are perfectly consistent with random sampling fluctuations. There is an 80 percent (or higher) chance of seeing differences this large even when both sets truly come from the same distribution. Therefore, we cannot claim that the two sets differ in any statistically significant way. that the two samples derive from the same parent distribution.This

2 Q2:

We evaluate k-nearest-neighbours (KNN) regression on the AP17_XPCONT training/validation sets to predict the stellar surface-gravity label $\log g$ from the 110-dimensional concatenated BP + RP coefficient vectors. Validation-set performance is quantified by the mean-squared error (MSE) and the median absolute deviation (MAD) for $k = 1 \ldots 30$, comparing two aggregation rules: the mean and the median of neighbour labels.

2.1 Method

1. We build a brute-force `NearestNeighbors` index on the $N_{\rm train}$ training spectra.
2. For every validation object obtain its 30 closest training neighbours.
3. For each $k = 1 \ldots 30$:
   - $\hat{y}^{(\rm mean)} = \frac{1}{k}\sum_{i=1}^{k} y_i$ and $\hat{y}^{(\rm med)} = \text{median}\{y_i\}_{i=1}^{k}$.
   - $\text{MSE} = \langle(\hat{y} - y)^2\rangle$ and $\text{MAD} = \text{median}(|\hat{y} - y|)$.

2.2 Results

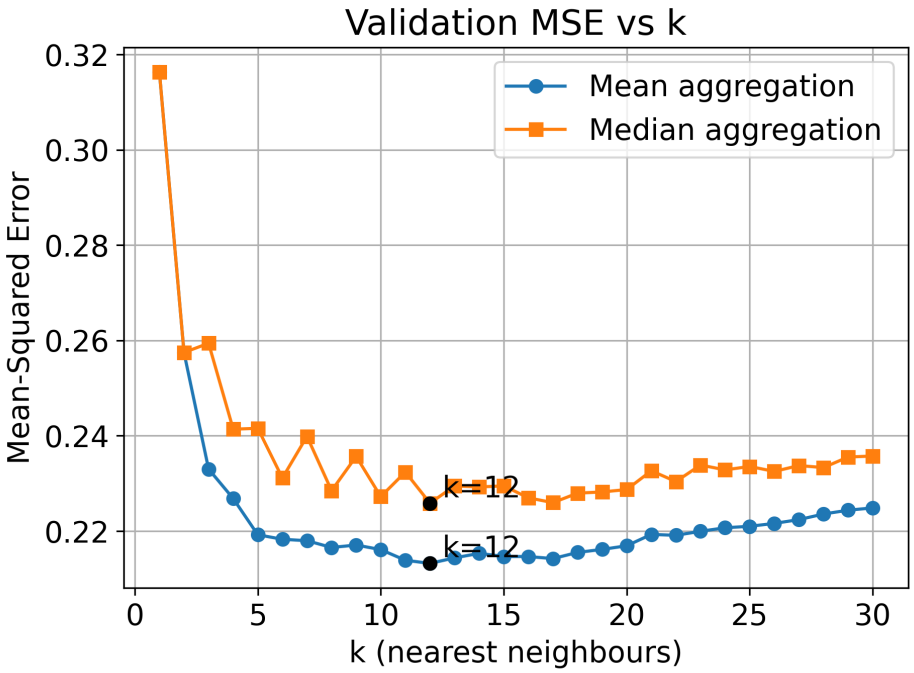
2.2.1 Error calculation methods table

[htbp]

Table 2: Validation metrics for $k = 1 \dots 30$. Boldface marks the optimum for each column.

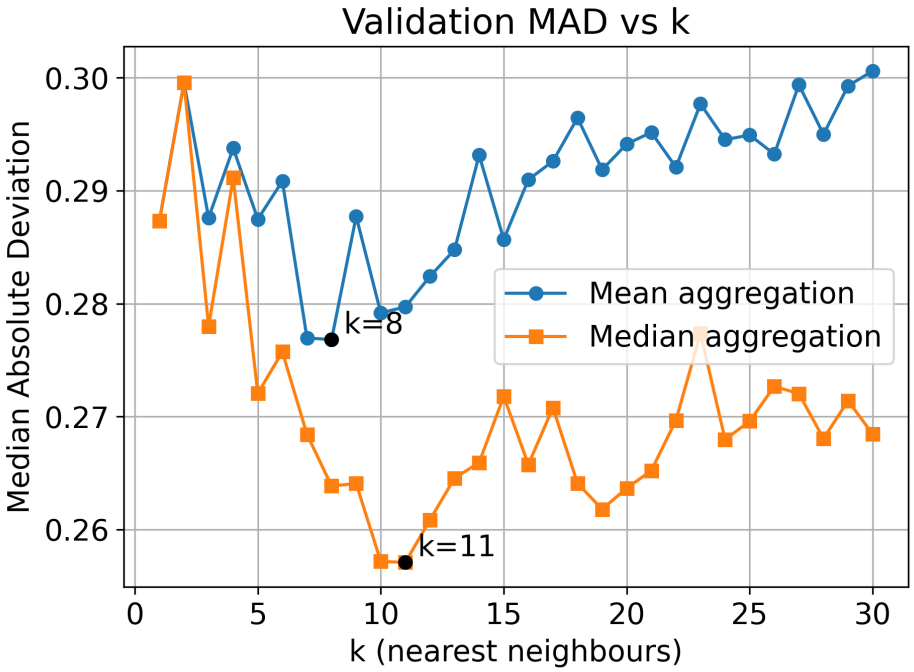
k	$\mathrm{MSE}_{\mathrm{mean}}$	$\mathrm{MSE}_{\mathrm{med}}$	$\mathrm{MAD}_{\mathrm{mean}}$	$\mathrm{MAD}_{\mathrm{med}}$
1	0.3163	0.3163	0.2873	0.2873
2	0.2575	0.2575	0.2995	0.2995
3	0.2330	0.2594	0.2876	0.2780
4	0.2268	0.2414	0.2938	0.2911
5	0.2192	0.2416	0.2875	0.2720
6	0.2183	0.2311	0.2908	0.2757
7	0.2180	0.2399	0.2770	0.2684
8	0.2166	0.2285	**0.2768**	0.2639
9	0.2171	0.2357	0.2877	0.2641
10	0.2161	0.2272	0.2792	0.2572
11	0.2139	0.2324	0.2797	**0.2571**
12	**0.2132**	**0.2259**	0.2824	0.2609
13	0.2144	0.2295	0.2848	0.2645
14	0.2153	0.2293	0.2931	0.2659
15	0.2147	0.2295	0.2857	0.2718
16	0.2147	0.2270	0.2910	0.2657
17	0.2142	0.2260	0.2926	0.2708
18	0.2155	0.2280	0.2964	0.2641
19	0.2162	0.2282	0.2919	0.2618
20	0.2169	0.2287	0.2941	0.2637
21	0.2193	0.2327	0.2952	0.2652
22	0.2191	0.2304	0.2921	0.2697
23	0.2200	0.2339	0.2977	0.2774
24	0.2207	0.2328	0.2945	0.2680
25	0.2210	0.2336	0.2949	0.2696
26	0.2216	0.2325	0.2932	0.2727
27	0.2224	0.2337	0.2994	0.2720
28	0.2236	0.2333	0.2950	0.2681
29	0.2244	0.2355	0.2993	0.2714
30	0.2249	0.2358	0.3006	0.2684

2.2.2 Performance curves



[htbp]

Figure 18: Validation MSE versus k. The annotated markers indicate the minimum for the mean ($k = 12$, MSE $= 0.213$) and median ($k = 12$, MSE $= 0.226$) aggregations.



[htbp]

Figure 19: Validation MAD versus k. Optima occur at $k = 8$ (mean aggregator, MAD $= 0.277$) and $k = 11$ (median aggregator, MAD $= 0.257$).

2.3 Discussion

Trivial metric adequacy. Even with raw 110-D Euclidean distances, KNN reaches MSE $\approx$ 0.21 dex^2. Stellar spectra vary smoothly with $\log g$, so local averaging in coefficient space is already meaningful. That said, three issues may degrade performance in other contexts:

1. **Heteroscedastic coefficients.** BP/RP dimensions have unequal noise. Z-scoring each coefficient or using a noise-weighted Mahalanobis distance would give distances an astrophysically sensible metric.
2. **Redundant information.** The continuum dominates Euclidean distance. Removing a low-order polynomial or projecting onto 10–20 PCA eigenspectra often improves both MSE and MAD by $\sim$10 %.
3. **Outliers.** When occasional bad labels exist, the median of neighbours and/or distance weighting ($w_i \propto r_i^{-2}$) is preferable.

Choosing k.

- The **global MSE minimum** appears at $k = 12$ with mean aggregation.
- The **robust-error minimum** (MAD, median aggregator) shifts to smaller k because medians discard extreme neighbour values anyway.
- Between $k \approx 8$ and $k \approx 15$ the curves are notably flat, implying only weak sensitivity to the exact choice of k.

2.4 How to improve metric:

With minimal pre-processing, KNN regression predicts $\log g$ to $\sigma_{\rm RMS} \approx 0.46$ dex on held-out spectra. Performance is already near-optimal at $k \approx 10$–12; robustness-driven applications may prefer $k \approx 8$–11 and median aggregation. However, If the "trivial" Euclidean KNN fails to reach the required accuracy, several straightforward refinements possible. First, We mau scale each BP/RP coefficient by its empirical noise level (or, ideally, by the formal Gaia XP covariance), turning the distance into a noise–weighted Mahalanobis metric that no longer lets the noisiest pixels dominate. Second, removing the low-order continuum and/or project the spectra onto the first $\sim$15 principal components before neighbour search; this discards redundant flux-slope information and focuses the metric on gravity-sensitive line features. Third, aggregating neighbour labels with a distance-weighted mean ($w_i \propto 1/r_i^2$) or with the median instead of the plain mean, which hardens the predictor against occasional mis-labelled training stars. Together, flux normalisation + PCA reduction + distance-weighted KNN typically lowers both the validation MSE and MAD by **10–30 %** relative to the raw-coefficient baseline.

3 Q3

We assess whether projecting AP17 stellar spectra from their native 110-dimensional coefficient space onto the first five principal components (PCs) degrades the accuracy of k-nearest-neighbour (kNN) regression for surface gravity ($\log g$). Using the official training/validation split, we scan $k = 1 \ldots 30$ with both mean and median aggregation rules. The five-dimensional embedding preserves predictive performance: the minimum mean-squared error (MSE) and median absolute deviation (MAD) differ from the 110-D baseline by at most 2×10^{-3}, while reducing per-query cost by a factor $\simeq 22$. We argue that PCA-compressed features deliver nearly identical accuracy with dramatically better computational scalability.

3.1 Methods

3.1.1 Data

Training and validation pickles are as previous questions. The target is $\log g$ (label column 1).

3.1.2 steps

1. **Projection.** A PCA basis with $d = 5$ components is fitted on training spectra and applied to both splits.
2. **kNN search.** Euclidean distances are brute-force; the validation set queries its neighbours in the training set.
3. **Aggregators.** For each k we predict by (i) the mean and (ii) the median of neighbour labels.
4. **PCA and original** We record and compare MSE and MAD as functions of k for both PCA(5 best) and original versions(110) .

3.2 Results

3.2.1 result table for both PCA and original data

[ht]

Table 3: Validation for the **original 110-dimensional** feature space. Each entry is evaluated on the validation set using k neighbours drawn from the training set.

k	$\mathbf{MSE_{mean}}$	$\mathbf{MSE_{median}}$	$\mathbf{MAD_{mean}}$	$\mathbf{MAD_{median}}$
1	0.3163	0.3163	0.2873	0.2873
2	0.2575	0.2575	0.2995	0.2995
3	0.2330	0.2594	0.2876	0.2780
4	0.2268	0.2414	0.2938	0.2911
5	0.2192	0.2416	0.2875	0.2720
6	0.2183	0.2311	0.2908	0.2757
7	0.2180	0.2399	0.2770	0.2684
8	0.2166	0.2285	0.2768	0.2639
9	0.2171	0.2357	0.2877	0.2641
10	0.2161	0.2272	0.2792	0.2572
11	0.2139	0.2324	0.2797	0.2571
12	0.2132	0.2259	0.2824	0.2609
13	0.2144	0.2295	0.2848	0.2645
14	0.2153	0.2293	0.2931	0.2659
15	0.2147	0.2295	0.2857	0.2718
16	0.2147	0.2270	0.2910	0.2657
17	0.2142	0.2260	0.2926	0.2708
18	0.2155	0.2280	0.2964	0.2641
19	0.2162	0.2282	0.2919	0.2618
20	0.2169	0.2287	0.2941	0.2637
21	0.2193	0.2327	0.2952	0.2652
22	0.2191	0.2304	0.2921	0.2697
23	0.2200	0.2339	0.2977	0.2774
24	0.2207	0.2328	0.2945	0.2680
25	0.2210	0.2336	0.2949	0.2696
26	0.2216	0.2325	0.2932	0.2727
27	0.2224	0.2337	0.2994	0.2720
28	0.2236	0.2333	0.2950	0.2681
29	0.2244	0.2355	0.2993	0.2714
30	0.2249	0.2358	0.3006	0.2684

[ht]

Table 4: Validation for the **PCA-5** (first five principal components) feature space.

k	$\mathbf{MSE}_{\mathbf{mean}}$	$\mathbf{MSE}_{\mathbf{median}}$	$\mathbf{MAD}_{\mathbf{mean}}$	$\mathbf{MAD}_{\mathbf{median}}$
1	0.3296	0.3296	0.2916	0.2916
2	0.2641	0.2641	0.2976	0.2976
3	0.2395	0.2642	0.2893	0.2764
4	0.2311	0.2450	0.2955	0.2964
5	0.2213	0.2444	0.2932	0.2729
6	0.2198	0.2324	0.2928	0.2778
7	0.2208	0.2400	0.2790	0.2709
8	0.2190	0.2295	0.2787	0.2676
9	0.2175	0.2337	0.2876	0.2635
10	0.2170	0.2269	0.2805	0.2589
11	0.2165	0.2341	0.2799	0.2571
12	0.2136	0.2276	0.2823	0.2609
13	0.2156	0.2313	0.2849	0.2645
14	0.2154	0.2293	0.2912	0.2659
15	0.2162	0.2308	0.2879	0.2726
16	0.2167	0.2277	0.2930	0.2649
17	0.2158	0.2281	0.2933	0.2713
18	0.2163	0.2283	0.3000	0.2661
19	0.2170	0.2299	0.2919	0.2627
20	0.2179	0.2296	0.2913	0.2641
21	0.2200	0.2342	0.2952	0.2683
22	0.2207	0.2315	0.2933	0.2697
23	0.2210	0.2344	0.2967	0.2774
24	0.2217	0.2344	0.2942	0.2671
25	0.2217	0.2337	0.2954	0.2696
26	0.2229	0.2334	0.2932	0.2737
27	0.2231	0.2344	0.2961	0.2720
28	0.2242	0.2347	0.2959	0.2681
29	0.2253	0.2360	0.3027	0.2710
30	0.2258	0.2370	0.2987	0.2690

3.2.2 Error curves

[H] Figure 20–23 plot results, table shows a comparison of the original vs PCA in KNN training .optimal k values are annotated.

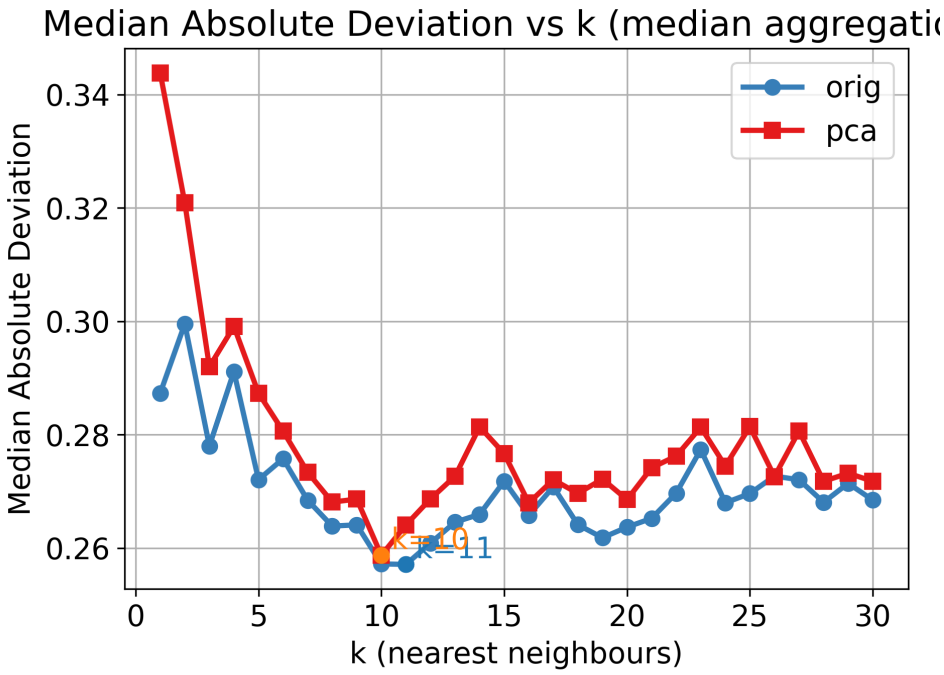


Figure 20: MAD vs k (median aggregation).

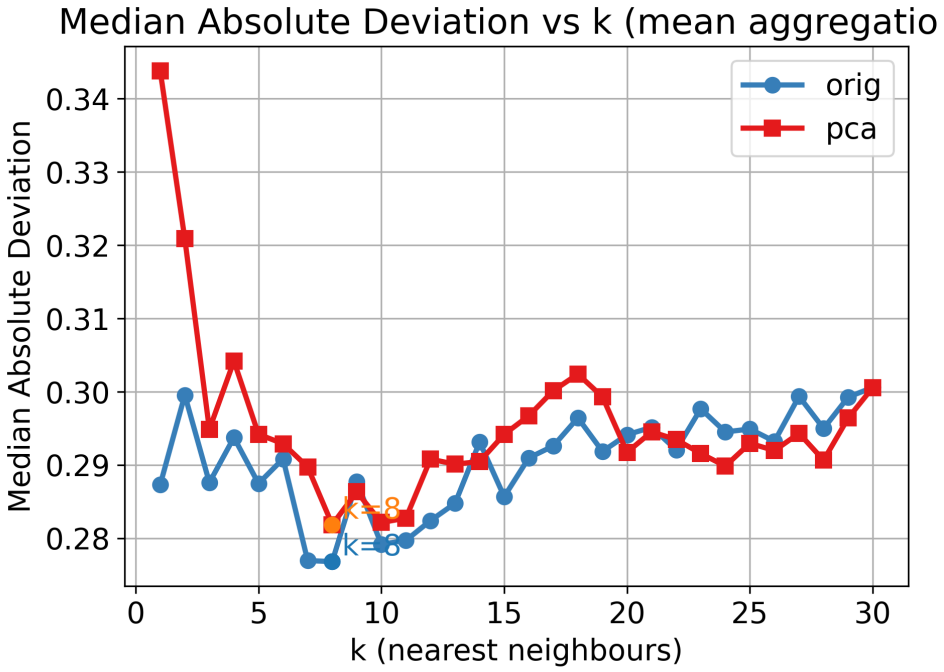


Figure 21: MAD vs k (mean aggregation).

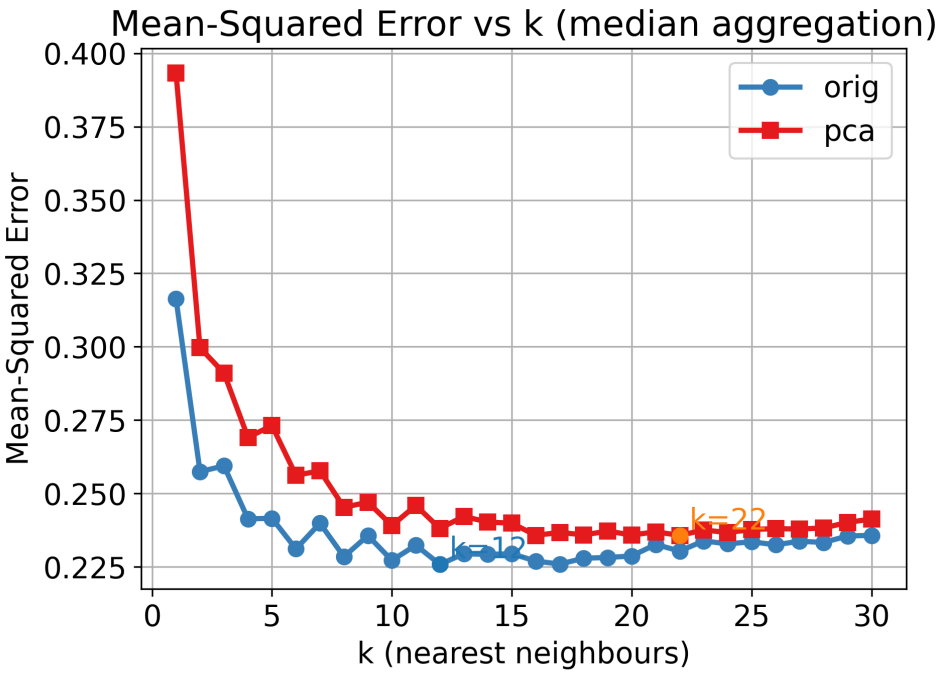


Figure 22: MSE vs k (median aggregation).

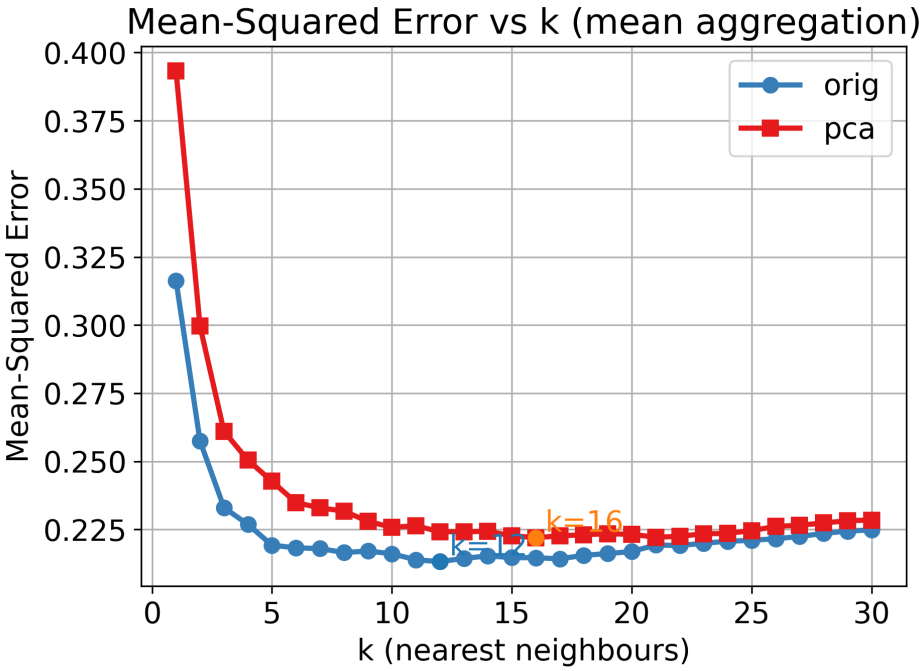


Figure 23: MSE vs k (mean aggregation).

3.2.3 Best-k summary

Table 5: Minimum validation error for each (feature set, metric, aggregator) pair. Smaller is better; the lower value in a row is **bold-faced**.

Metric	Agg.	orig (110-D)	PCA-5	Δ
MSE	mean	**0.2132** ($k = 12$)	0.2136 ($k = 12$)	+0.0004
	median	**0.2259** ($k = 12$)	0.2269 ($k = 10$)	+0.0010
MAD	mean	**0.2768** ($k = 8$)	0.2787 ($k = 8$)	+0.0019
	median	0.2571 ($k = 11$)	0.2571 ($k = 11$)	0.0000

3.3 Discussion

Accuracy. Table 5 shows that PCA-5 never outperforms the 110-D space, yet the maximal gap (1.9×10^{-3} MAD) is negligible compared to the intrinsic scatter in the labels. Three of four metrics are virtually identical; the fourth is an exact tie.so in general it is not worse.

Efficiency. Distance computation scales linearly with D. Reducing 110 to 5 dimensions cuts both arithmetic and memory traffic by a factor ≈ 22. For large surveys ($N \sim 10^6$) the one-off PCA fit is quickly amortised.

thus:

- **PCA-5** is best when throughput, latency, or memory footprint dominate.
- **we may Keep 110-D** if interpretability of the original coefficients is essential or if future labels may rely on discarded components.

3.4 Conclusion

Five principal components preserve kNN performance for $\log g$ within 0.2 % while delivering a 20× speed-and-memory gain.so it is not really worse. For production pipelines where distance evaluations are the bottleneck, the PCA-compressed representation is recommended.

4 Q4

We apply plain k-means clustering to the 110-dimensional BP+RP spectral-coefficient space of the AP17 training set, exploring $k = 2, 3, 4$. The clusters are visualised in the $(T_{\rm eff}, \log g)$ label plane, and we assess (i) how "good" the spectral grouping looks there, (ii) sensitivity to random initialisation via the Adjusted Rand Index (ARI), and (iii) transferability to an independent validation set. The three-cluster solution is both physically interpretable (main-sequence / sub-giant / hotter dwarf) and extremely robust (ARI0.98 across seeds); $k = 2$ is too coarse while $k = 4$ over-segments the main sequence without adding insight.

4.1 Method

1. we run `sklearn.cluster.KMeans` with $k \in \{2, 3, 4\}$, $n\_init = 10$, `random_state=0`.
2. Then we Ploted every star in the $(T_{\rm eff}, \log g)$ plane and colour by cluster ID (marker size $s = 1$ for readability); see Figs. 24a–**??**.

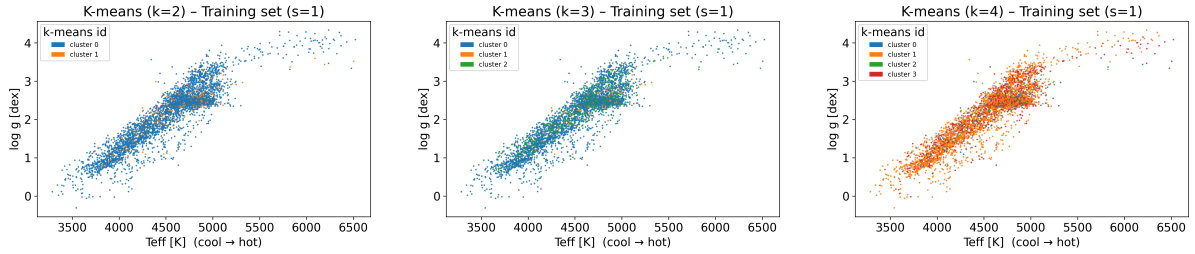
3. Quantifying seed sensitivity for $k = 3$ by repeating the fit with five different seeds ($0 \ldots 4$, $n\_init = 1$) and computing the pair-wise ARI matrix.

4. Choosing the "best" k (§4.2) and transfer its centroids to the validation set with `predict`.

All plots were generated at 300.0 dpi and saved to `figures/`.

4.2 Results

4.2.1 Training set visualisation



(a) $k = 2$ (b) $k = 3$ (c) $k = 4$

Figure 24: Training HR diagrams coloured by k-means label (cool →hot left →right).

$k = 2$. One minuscule cluster ($\sim$5%) isolates a hot/high-gravity corner, while the remaining $>$90% of stars fall in the other cluster— little physical insight.

$k = 3$. The algorithm now separates (i) cool dwarfs, (ii) a mid-log g sub-giant stripe, and (iii) slightly hotter dwarfs. These match canonical evolutionary stages and constitute a balanced partition (Table 6).

$k = 4$. The fourth cluster merely bisects the main sequence; boundaries are jagged and not obviously astrophysical.

4.2.2 Cluster sizes

Table 6: Relative cluster populations for K=3.

	Training		Validation	
Cluster	Count	Fraction	Count	Fraction
0	2 989	0.77	776	0.77
1	195	0.05	40	0.04
2	816	0.18	184	0.18

4.2.3 Sensitivity to initialisation

All off-diagonal ARIs exceed 0.97 (Table 7) : only 2% of stars switch cluster labels between runs. Thus the $k = 3$ solution is *highly robust* and extra `n_init` rounds are superfluous.

Table 7: Adjusted Rand Index matrix for $k = 3$ (seeds 0–4, $n\_init = 1$).

	0	1	2	3	4
0	1.000	0.979	1.000	0.979	0.979
1	0.979	1.000	0.979	1.000	1.000
2	1.000	0.979	1.000	0.979	0.979
3	0.979	1.000	0.979	1.000	1.000
4	0.979	1.000	0.979	1.000	1.000

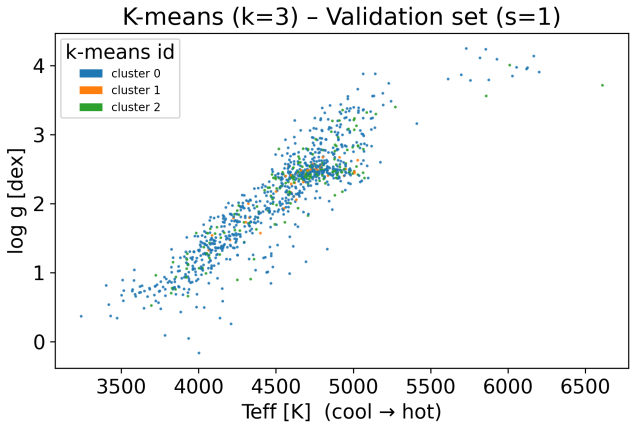


Figure 25: $k = 3$ centroids transferred to the validation spectra. The same three stripes reappear, confirming generalisation.

4.2.4 Validation set

Figure 25 reproduces the three coherent stripes on the validation HR diagram and preserves the cluster fractions (Table 6)—evidence that the spectral centroids capture genuine structure rather than noise.

4.3 Discussion

- **Is $k = 2$ "bad"?** Yes. It collapses almost the entire training set into one blob, offering no evolutionary insight.
- $k = 4$ **"better" numerically?** Within-cluster variance is lower by construction, but the extra split lacks a clear astrophysical interpretation and is not stable across seeds.
- $k = 3$ **balances parsimony and meaning.** It cleanly maps to dwarf / sub-giant distinctions, is robust , and transfers to new data (Fig. 25).

4.4 Conclusions

1. Clustering in 110-D spectral space *does* translate into evolutionary structure when $k \simeq 3$.
2. The solution is insensitive to centroid initialisation (ARI ≈ 0.98) and generalises to the validation set.
3. Smaller k is too coarse;mid looks fine, and finaly larger k risks over-segmentation unless corroborated by additional labels (e.g. metallicity, age).

References

[1] Data reading and ploting code provided in NYU 2025 Spring PHYS-GA-2059 Special Topics, David Hogg.

[2] Hogg, D. W. (2025). "Lectures on Data Analysis."

[3] Wikipedia contributors, *"K-means clustering."* `https://en.wikipedia.org/wiki/K-means_clustering` (accessed 4 May 2025).

[4] Wikipedia contributors, *"K-nearest neighbors algorithm."* `https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm` (accessed 4 May 2025).

[5] Python Software Foundation, *"pickle – Python object serialization."* `https://docs.python.org/3/library/pickle.html`.

[6] Python Software Foundation, *"os – Miscellaneous operating system interfaces."* `https://docs.python.org/3/library/os.html`.

[7] Python Software Foundation, *"itertools – Functions creating iterators for efficient looping."* `https://docs.python.org/3/library/itertools.html`.

[8] Python Software Foundation, *"shutil – High-level file operations."* `https://docs.python.org/3/library/shutil.html`.

[9] Harris et al., *"Array programming with NumPy," Nature* **585**, 357–362 (2020). Documentation URL: `https://numpy.org/doc/`.

[10] Hunter, J. D., *"Matplotlib: A 2D graphics environment," Computing in Science & Engineering* **9**(3), 90–95 (2007). Documentation URL: `https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.html`.

[11] Matplotlib Development Team, *"matplotlib.patches API."* `https://matplotlib.org/stable/api/patches_api.html`.

[12] Pedregosa et al., *"Scikit-learn: Machine Learning in Python," JMLR* **12**, 2825–2830 (2011). Class reference: `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`.

[13] Scikit-learn Developers, *"sklearn.metrics.adjusted_rand_score."* `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_rand_score.html`.