# **Tanisha’s Contribution — Advanced Statistics & Visualization**

## **1) What I set out to do**

I led the advanced EDA and visualization workstream with one clear aim: turn the raw dataset into evidence that decision-makers can act on. That meant going beyond pretty charts—designing a sequence of analyses that explain *what drives Food Insecurity (FI)*, *where it concentrates*, and *which levers matter most*.

## **2) How I approached the problem**

* **Analytic framing, not just plotting.** I converted ambiguous ideas into researchable questions (e.g., *“Do regions with higher unsheltered composition experience disproportionate FI?”*, *“Which states are persistent risk clusters across years?”*, *“Which counties are hotspots even after controlling for fundamentals?”*).
* **Robust, reproducible workflow.** I wrote cells that (a) coerce numeric types safely, (b) trim extreme outliers for readability without hiding structure, and (c) guard against common errors (e.g., column name drift like Num\_restaurants vs Num Restaurants).
* **Visuals built for decisions.** Every key scatter includes an average binned-trend overlay so leaders can see the *shape* of the relationship (thresholds, curvature) instead of a noisy cloud. Figures render inline; each prints a quick stat (e.g., correlation) right below the chart to anchor the takeaway.

## **3) What I delivered**

* **Unemployment is the biggest signal.**I made a clean chart that shows: as **unemployment goes up, food insecurity also goes up**. The relationship is steady and strong (about **+0.61 correlation**). Takeaway: **jobs and earnings** are the most important levers to reduce food insecurity.
* **Unsheltered vs. sheltered homelessness.** I compared food insecurity with both **unsheltered** and **sheltered** homelessness and also created an **“unsheltered share”** measure. Results: areas with **more unsheltered people** have **higher food insecurity**, and this increases step by step as the unsheltered share rises. When we put both in a small model, **unsheltered** matters more than sheltered. Takeaway: **being unsheltered is a stronger risk factor** and needs targeted housing and outreach.
* **Costs and access—read correctly.**
  + **Rent:** In our data, higher rent is linked to **lower** food insecurity. That doesn’t mean “high rent is good.” It likely means **wealthier places have both higher rents and better resources**. I called this out to avoid confusion and recommended looking at rent **together with** income, unemployment, and **Cost Per Meal** (not by itself).
  + **Food outlet counts:** Just counting **grocery stores** or **restaurants** doesn’t explain much by itself. Counts don’t capture **price, distance, or quality**. I suggested using these as **secondary features** or in **interactions** (e.g., with population or income) instead of treating them as stand-alone indicators.
* **Where risk is concentrated (place-based view).** I built a **Hotspot Index** combining unemployment, houselessness, and food insecurity (standardized to be comparable). This ranked **counties with stacked problems** and showed the worst cases in a simple heat chart for leaders to scan quickly.
* **Which states keep showing up as high risk.** I checked how often each state lands in the **top 20% of food insecurity** over the years. Several **Southern states** show up again and again, which points to **structural (long-term) issues** rather than one-time spikes. These areas need **sustained** investment, not one-off fixes.
* **Beyond the usual factors (residual hotspots).** I fit a simple model using the main drivers:  
   **FI = Unemployment + Houselessness + Unsheltered + Rent + Cost Per Meal**.  
   Then I looked at **residuals**—places where **actual food insecurity is higher than what the model predicts**. Those places likely have **extra barriers** (e.g., program access, transport, local food markets) and should be **priority targets** for on-the-ground investigation. I also flagged **positive outliers** (doing better than expected) so we can learn what’s working there.

## **4) What I built that others can reuse**

* **Engineered features** for stronger signal and cleaner stories:  
  + unsheltered\_share = Unsheltered / (Unsheltered + Sheltered)
  + grocery\_per\_10k = Num\_grocery / TOT\_POP × 10,000
  + rest\_to\_grocery\_ratio = Num\_restaurants / (Num\_grocery + ε) with safe ε to avoid divide-by-zero
  + cost\_pressure\_index = mean(z(Rent), z(Cost Per Meal))
  + simple time features (Year, YoY deltas) + **state indicators** for fixed-effects modeling
* **Reusable plotting helpers** (binned mean trend overlays) that convert any scatter into a decision graphic.
* **Defensive utilities** for column canonicalization (solves KeyErrors from spaces/case), so analyses won’t break when data pipelines change.

## **5) Challenges I solved (so the work “just works”)**

* **Column inconsistencies & hidden characters.** I implemented a canonical mapping to handle Num\_restaurants vs Num Restaurants vs trailing spaces.
* **Exploding ratios from zero denominators.** I stabilized with ε and winsorized the 1st–99th percentile for readable charts without distorting central patterns.
* **Weak univariate proxies.** Instead of forcing a story, I documented limits and reframed access and cost metrics within **multivariate** and **interaction-aware** modeling.

# Contribution Report — Asad Naqvi & Maleesha

(County-level pipeline & prediction, aligned with the reference project’s data/model structure)

## 1) Objective

Deliver a county-level (FIPS) panel covering 2010–2023 and a prediction workflow that mirrors the reference repository’s structure—so results are comparable, leakage-safe, and immediately usable by EDA, modelling, and mapping teams.

## 2) Data Pre-processing (fully repo-aligned)

### Unit & keys

County FIPS (padded to 5) and Year (integer), matching the reference granularity and time grain.

### Schema & hygiene

Headers canonicalised to snake\_case; consistent feature names across years.

Value cleaning: removed %/commas; mapped --/NA/N/A/./"" → nulls; safe numeric coercion.

Year handling: derived year from date fields when needed; annualised repeated period rows to a single (fips, year) record (numeric mean).

Per-capita metrics: constructed \*\_per\_10k where total\_pop was available for fair cross-county comparison.

### Integration & outputs

Outer joins on [fips, year] to preserve coverage; sorted and de-duplicated.

Produced two master panels with identical schemas:

panel\_county\_2010\_2023\_clean.csv — NA-preserved (split-agnostic).

panel\_county\_2010\_2023\_imputed.csv — simple imputation (numeric=median; categorical=mode) for quick EDA.

### Documentation & diagnostics:

panel\_county\_2010\_2023.data\_dictionary.csv (dtype, pre-imputation missing%).

feature\_coverage\_2010\_2023.csv (per-year & overall availability).

per\_year/county\_YYYY.csv (snapshot exports).

Known data realities transparently handled

FI targets 2011–2013 available only at state level in the provided sources—flagged accordingly and excluded from county-level target training unless a state→county backfill is explicitly approved.

Partial coverage for some feature families (e.g., business counts) surfaced via the coverage matrix and tagged “Add-on,” while high-coverage features were tagged “Core.”

## 3) Prediction Workflow (same structure as the reference model)

Problem

Forecast county FI Rate for a future year.

Split & leakage control

Train: 2010–2022

Test (hold-out): 2023 (true out-of-time evaluation)

Preprocessing (imputation, scaling, encoding) fit on train only, then applied to test.

Key persistence feature

FI\_prev = prior-year FI within county (constructed by sorting within FIPS and shifting before the time split), reflecting the well-documented persistence: “this year ≈ last year + adjustments.”

Baselines & validation

Baselines: state-mean and carry-forward (t−1) to contextualise model lift.

Validation: GroupKFold by state to reduce geographic leakage (no state overlaps across folds).

Models (aligned family & rationale)

OLS + FI\_prev as the transparent primary model (interpretable coefficients).

Regularised/feature-reduced linear (multicollinearity control).

Non-linear checks (Random Forest / XGBoost / CatBoost) to probe interactions and curvature.

Prediction deliverables

Fitted preprocessing artifacts (encoders/scalers/imputers), trained model snapshot(s), and 2023 hold-out predictions (fips, year, y\_true, y\_pred).

A concise model card (features used, split policy, CV scheme, assumptions, limitations).

## 4) Reusable Assets

Header canonicaliser & column resolver (handles spaces/case/hidden characters).

Numeric/type coercers with robust token-to-NA mapping.

Year derivation utilities and an annualiser for monthly/periodic inputs.

Per-capita constructor: metric\_per\_10k = metric / total\_pop \* 10\_000.

Join orchestrator for outer merges on [fips, year].

Coverage matrix generator to support Core vs Add-on feature selection without editing the pipeline.

## 5) Challenges Solved

Column drift across sources and years → unified schema that prevents downstream key errors.

Zero denominators / extreme ratios → safe epsilons and light winsorisation in EDA contexts.

Patchy temporal coverage → surfaced, documented, and separated into Core vs Add-on; panel remains consistent.

Team interoperability → shared both clean (NA-kept) and imputed panels; no split baked into data (modelling applies its split, matching the reference structure).

## 6) Alignment & Intentional Differences

Aligned: geo-time keys, cleaning steps, year stacking, horizontal merges, rate/per-capita logic, leakage-safe modelling and evaluation.

Intentional choices: explicitly county 2010–2023 to match the modelling report (train ≤2022; test 2023) while keeping the shared datasets split-agnostic.

## 7) Outcome

A transparent, reproducible path from raw county datasets to leakage-safe 2023 predictions, with artefacts that downstream teams can trust and reuse: clean & imputed panels, data dictionary, coverage diagnostics, and model outputs—all structured to match the reference project’s modelling and evaluation approach.

# **Modeling Report — Dona**

# **1) Problem framing (what we’re predicting)**

* **Target:** County-level Food Insecurity rate (**FI Rate**) for a future year.
* **Unit:** County (FIPS), multiple years (panel data).
* **Evaluation year:** **2023** is held out as a **true future** test set.
* **Train window:** **2010–2022**.

Why this matters: if we evaluate on 2023 after training on 2010–2022 only, we get a realistic measure of how well we’ll do when we deploy to the next year.

## **2) What “t−1” / FI\_prev means, and why it’s central**

* **Definition:** FI\_prev = the **previous year’s FI rate** for the **same county**.  
   Formally: for county *i* in year *t*, FI\_prev(i,t) = FI(i,t−1).
* **How it’s built (in code):**
  + Sort rows by FIPS, then Year.
  + Within each FIPS, **shift the target** one year down to create FI\_prev.
  + After that, perform the **time-aware split** (train: ≤2022; test: 2023).
* Why this is **not leakage**:  
   When we predict **year T** in the real world, **year T−1 FI** is already known (it’s last year’s published estimate). Using FI\_prev mirrors how we’d actually forecast: *tomorrow is a lot like today*, then adjust using current fundamentals (unemployment, costs, etc.).
* **Why it’s important:** FI is **highly persistent** year to year. FI\_prev captures this persistence in one interpretable feature, which:  
  + Boosts accuracy dramatically,
  + Keeps the model simple and explainable,
  + Lets us add **incremental** drivers (unemployment, cost per meal) on top.

tl;dr: We model **“this year = last year + adjustments”**. That’s what FI\_prev operationalizes.

## **3) Data hygiene steps inside the notebook**

* **Column normalization:** FIPS padded to 5 chars, STATE from first two FIPS digits, Year to int.
* **Row filtering:** keep only rows with a non-missing **FI Rate**.
* **Sparse-feature pruning:** drop features with **>60% missingness in the train window** (prevents peeking at test-year missingness patterns).
* **Time-aware split:**
  + Train = **2010–2022**
  + Test = **2023**
* **Baselines created:**
  + **State mean:** avg FI by state over 2010–2022.
  + **Carry-forward (t−1):** use FI\_prev if available; else fallback to state mean.
* **Preprocessing (sklearn Pipeline + ColumnTransformer):**
  + **Numerics:** median imputation **+ missingness indicators** (so “was missing” can be predictive) and **scaling** (for linear models).
  + **Categoricals:** One-Hot Encoding with handle\_unknown="ignore".
  + Important: all preprocessing is **fit on train only**, then applied to test.
* **Grouped cross-validation:** **GroupKFold by STATE** (5 folds) to avoid geographic leakage (counties from the same state don’t leak across train/valid).S

## **4) Models trained (and what each is for)**

**Linear**

* **M1 — OLS (all features) + FI\_prev** Simple, fast, fully explainable via coefficients.
* **M2 — OLS (VIF≤10 numerics) + FI\_prev** Reduces multicollinearity for stabler coefficients.

**Nonlinear**

* **M3 — XGBoost + FI\_prev** Captures interactions/nonlinearities.
* **M4 — CatBoost + FI\_prev**
* **M5 — Random Forest + FI\_prev**

**Hybrid**

* **M6 — Simple stack (OLS + boosting) with state-residual correction** Blend of models; adjust predictions by the average residual per state learned on train.

All six use the **same preprocessing** and are evaluated on:

* **2023 hold-out**, and
* **5-fold GroupKFold by STATE** within the 2010–2022 train window.

## **5) What won (and what that means)**

With FI\_prev included:

* **M1 — OLS(+FI\_prev)** delivered the **best combination** of **accuracy + interpretability + stability**:  
  + **2023 test:** **R² ≈ 0.995**, **RMSE ≈ 0.0025**
  + **CV (5-fold by state on train years):** **R² ≈ 0.974 ± 0.014**, **RMSE ≈ 0.0060 ± 0.0013**
* Tree/boosting models and the hybrid stack were close but didn’t materially beat OLS on test while being less transparent.

**Interpretation:**

* **FI\_prev** is the dominant driver (FI is persistent).
* **Unemployment rate** and **Cost per Meal** add intuitive, incremental lift.
* Coefficients provide a clear story that aligns with our EDA.

**Production choice:** **Use OLS(+FI\_prev)** for forecasting; keep XGB/CatBoost as sanity checks.

## **6) What’s inside the notebook outputs (so you know what you’re seeing)**

* **Baseline metrics** for state-mean and carry-forward (context).
* **Per-model metrics** (train R²/RMSE, 2023 test R²/RMSE, state-grouped CV).
* **Coefficient table** for OLS(+FI\_prev) showing feature signs/magnitudes.

## **7) Guardrails & caveats**

* This setup is **purpose-built for next-year forecasts** (where FI\_prev is known).
* It **avoids leakage** by computing FI\_prev within-FIPS, then doing the time split, and by fitting preprocessing on train only.
* For counties with sparse covariates, **missing indicators** help but uncertainty is naturally higher.
* As policies and economic conditions shift, **monitor residuals over time** to detect drift.

## **8) Summary**

We forecast county FI rates using a transparent **OLS model with the prior year’s FI (FI\_prev)** plus core fundamentals (**unemployment**, **cost per meal**, key demographics). This is not leakage—last year’s FI is known when forecasting next year. The model strongly outperforms baselines and generalizes across states (**2023 R² ≈ 0.995; CV R² ≈ 0.97**). It is simple to run for new years, coefficients are interpretable, and choropleths/residual maps validate spatial patterns and flag counties where FI is higher than expected given fundamentals.