Below is the **full updated scorer** (with tuned weights + scarcity/holiday logic) and a **clear, locked-in spec** for the roadmap + spot score. Paste the code into scripts/spot\_value\_updates.py, then run the commands at the end.

# A) Updated scripts/spot\_value\_updates.py (drop‑in replacement)

#!/usr/bin/env python3  
# Survivor spot value scorer — v1.0 (LOCKED)  
# LIVE DVOA only (no projections) + scarcity + holiday highlights  
  
from pathlib import Path  
import sys, argparse  
import numpy as np, pandas as pd  
  
ROOT = Path(\_\_file\_\_).resolve().parents[1]  
if str(ROOT) not in sys.path:  
 sys.path.insert(0, str(ROOT))  
  
ROADMAP = ROOT / "picks" / "survivor" / "survivor\_roadmap\_expanded.csv"  
  
# ==========================  
# Tuned weights & thresholds  
# ==========================  
# Core components  
W\_WIN = 0.42  
W\_HOME = 0.12  
W\_REST = 0.10  
W\_RATING, RATING\_WIDTH = 0.12, 8.0  
W\_INJURY, INJURY\_CAP = 0.06, 0.10  
W\_SCARCITY\_TOTAL = 0.10  
  
# DVOA (LIVE snapshot)  
W\_DVOA\_LEVEL = 0.16 # was 0.12  
W\_DVOA\_TREND = 0.06 # was 0.05  
LEVEL\_CAP = 0.18 # was 0.15 (cap on abs level contribution in decimal)  
TREND\_SCALE\_PP = 10.0 # 10 pp ~ 1.0 unit  
MAX\_TREND\_BONUS= 0.03  
BAND\_BUMP = {"UP": 0.015, "DOWN": -0.012} # tiny categorical nudges  
  
# Bucket thresholds (fixed; no quantiles)  
HI\_THRESH = 0.52 # was 0.55  
MED\_THRESH = 0.42 # was 0.44  
  
# Holiday penalties (unchanged; applied only to holiday weeks)  
HOLIDAY\_TG\_PENALTY = -0.12  
HOLIDAY\_BF\_PENALTY = -0.08  
HOLIDAY\_XMAS\_PENALTY= -0.12  
HOLIDAY\_COMBO\_EXTRA = -0.05  
  
# Scarcity/Now-or-Never knobs  
NOW\_NEVER\_MARGIN = 0.05 # need +5pp vs best remaining week to qualify  
NOW\_NEVER\_BONUS = 0.02 # max bonus for clear last/best shot  
  
# Holiday highlight (UI guidance)  
HOLIDAY\_HOLDOUT\_SOFT = 0.00 # 0.00 = no score effect; set -0.02 to gently discourage early use  
  
  
def \_parse\_weeks(s):  
 if not s or str(s).lower() == "all":  
 return None  
 out = set()  
 for part in str(s).split(','):  
 part = part.strip()  
 if not part: continue  
 if '-' in part:  
 a,b = part.split('-',1)  
 out.update(range(int(a), int(b)+1))  
 else:  
 out.add(int(part))  
 return sorted(out)  
  
  
def ensure\_columns(df: pd.DataFrame) -> pd.DataFrame:  
 defaults = {  
 "projected\_win\_prob": 0.50,  
 "home\_or\_away": "Home",  
 "rest\_days": 6,  
 "rating\_gap": np.nan,  
 "injury\_adjustment": 0.0,  
 # holiday flags  
 "is\_thanksgiving": 0, "is\_black\_friday": 0, "is\_christmas": 0, "plays\_both\_tg\_xmas": 0,  
 # LIVE DVOA fields from compute\_dvoa\_trends.py  
 "dvoa\_gap\_dec": np.nan, # decimal gap of total DVOA (team - opp)/100  
 "trend3\_pp": np.nan, # EMA(3) delta in pp  
 "trend\_band": "", # Up / Flat / Down / Unknown  
 }  
 for k,v in defaults.items():  
 if k not in df.columns:  
 df[k] = v  
 # types  
 for c in ["projected\_win\_prob","rest\_days","rating\_gap","injury\_adjustment","dvoa\_gap\_dec","trend3\_pp"]:  
 df[c] = pd.to\_numeric(df[c], errors="coerce")  
 for c in ["is\_thanksgiving","is\_black\_friday","is\_christmas","plays\_both\_tg\_xmas"]:  
 df[c] = pd.to\_numeric(df[c], errors="coerce").fillna(0).astype(int)  
 if "week" in df.columns:  
 df["week"] = pd.to\_numeric(df["week"], errors="coerce").astype("Int64")  
 return df  
  
  
def win\_component(df: pd.DataFrame) -> pd.Series:  
 wp = df["projected\_win\_prob"].clip(0,1).fillna(0.5)  
 k = 6.0  
 return W\_WIN \* (1.0 / (1.0 + np.exp(-k\*(wp-0.5))))  
  
  
def base\_score(df: pd.DataFrame) -> pd.DataFrame:  
 df["sv\_win"] = win\_component(df)  
 df["sv\_home"] = (df["home\_or\_away"].str.lower()=="home").astype(float) \* W\_HOME  
 df["sv\_rest"] = ((df["rest\_days"].fillna(0).clip(4,10)-4)/6.0) \* W\_REST  
 df["spot\_value\_score"] = (df["sv\_win"] + df["sv\_home"] + df["sv\_rest"]).clip(0,1)  
 return df  
  
  
def add\_rating\_component(df: pd.DataFrame) -> pd.DataFrame:  
 g = df["rating\_gap"].fillna(0.0)  
 adj = W\_RATING \* np.tanh(g / RATING\_WIDTH)  
 df["sv\_rating"] = adj  
 df["spot\_value\_score"] = (df["spot\_value\_score"] + adj).clip(0,1)  
 return df  
  
  
def add\_dvoa\_component(df: pd.DataFrame) -> pd.DataFrame:  
 # 1) Level (LIVE)  
 level = df["dvoa\_gap\_dec"].fillna(0.0).clip(-LEVEL\_CAP, LEVEL\_CAP)  
 sv\_level = W\_DVOA\_LEVEL \* level  
  
 # 2) Trend (LIVE)  
 trend\_norm = (df["trend3\_pp"].fillna(0.0) / TREND\_SCALE\_PP).clip(-1.0, 1.0)  
 sv\_trend = (W\_DVOA\_TREND \* trend\_norm).clip(-MAX\_TREND\_BONUS, MAX\_TREND\_BONUS)  
 if "week" in df.columns:  
 early = df["week"].fillna(99).astype(int) < 5  
 sv\_trend = sv\_trend.where(~early, sv\_trend \* 0.4)  
  
 # 3) Band nudge  
 bump = df["trend\_band"].astype(str).str.upper().map(BAND\_BUMP).fillna(0.0)  
  
 df["sv\_dvoa\_level"], df["sv\_dvoa\_trend"], df["sv\_dvoa\_band"] = sv\_level, sv\_trend, bump  
 df["sv\_dvoa"] = sv\_level + sv\_trend + bump  
 df["spot\_value\_score"] = (df["spot\_value\_score"] + df["sv\_dvoa"]).clip(0,1)  
 return df  
  
  
def add\_injury\_component(df: pd.DataFrame) -> pd.DataFrame:  
 inj = df["injury\_adjustment"].fillna(0.0).clip(-INJURY\_CAP, INJURY\_CAP)  
 df["sv\_injury"] = W\_INJURY \* inj  
 df["spot\_value\_score"] = (df["spot\_value\_score"] + df["sv\_injury"]).clip(0,1)  
 return df  
  
  
def add\_holiday\_penalty(df: pd.DataFrame) -> pd.DataFrame:  
 tg = df["is\_thanksgiving"].astype(int)  
 bf = df["is\_black\_friday"].astype(int)  
 xm = df["is\_christmas"].astype(int)  
 both = df["plays\_both\_tg\_xmas"].astype(int)  
 base = tg\*HOLIDAY\_TG\_PENALTY + bf\*HOLIDAY\_BF\_PENALTY + xm\*HOLIDAY\_XMAS\_PENALTY  
 combo = (((tg+bf)>0)&(xm>0) | (both>0)&((tg+bf+xm)>0)).astype(int)\*HOLIDAY\_COMBO\_EXTRA  
 df["sv\_holiday"] = base + combo  
 df["spot\_value\_score"] = (df["spot\_value\_score"] + df["sv\_holiday"]).clip(0,1)  
 return df  
  
  
def add\_future\_scarcity\_boost(df: pd.DataFrame) -> pd.DataFrame:  
 # Also computes now-or-never bump and flags  
 def per\_team(g: pd.DataFrame) -> pd.DataFrame:  
 p = pd.to\_numeric(g["projected\_win\_prob"], errors="coerce").fillna(0.0).to\_numpy()  
 rev\_cummax = np.maximum.accumulate(p[::-1])[::-1]  
 max\_future = np.roll(rev\_cummax, -1); max\_future[-1] = 0.0  
 g = g.copy()  
 g["max\_future\_prob"] = max\_future  
 # scarcity (bonus only)  
 raw = (g["projected\_win\_prob"] - g["max\_future\_prob"]).clip(lower=0.0)  
 g["sv\_scarcity\_raw"] = (raw / 0.30).clip(0.0, 1.0)  
 g["sv\_scarcity"] = W\_SCARCITY\_TOTAL \* g["sv\_scarcity\_raw"]  
 # now-or-never  
 g["is\_now\_or\_never"] = g["projected\_win\_prob"] >= g["max\_future\_prob"] + NOW\_NEVER\_MARGIN  
 gap\_over = (g["projected\_win\_prob"] - g["max\_future\_prob"] - NOW\_NEVER\_MARGIN).clip(lower=0.0)  
 scale = (gap\_over / 0.10).clip(0.0, 1.0)  
 g["sv\_now\_or\_never"] = NOW\_NEVER\_BONUS \* scale  
 return g  
  
 df = df.groupby("team", group\_keys=False).apply(per\_team)  
 df["spot\_value\_score"] = (df["spot\_value\_score"] + df["sv\_scarcity"] + df["sv\_now\_or\_never"]).clip(0,1)  
 return df  
  
  
def add\_holiday\_highlights(df: pd.DataFrame) -> pd.DataFrame:  
 hol\_any = (df[["is\_thanksgiving","is\_black\_friday","is\_christmas"]].fillna(0).astype(int).sum(axis=1) > 0)  
 df["holiday\_any"] = hol\_any.astype(int)  
 anchor = (df[df["holiday\_any"] == 1].groupby("team")["week"].min())  
 df["holiday\_anchor\_week"] = df["team"].map(anchor).astype("Int64")  
 df["is\_holiday\_team"] = df["holiday\_anchor\_week"].notna()  
 df["suggest\_save\_for\_holiday"] = df["is\_holiday\_team"] & (df["week"].astype("Int64").fillna(99) < df["holiday\_anchor\_week"].astype("Int64"))  
 if HOLIDAY\_HOLDOUT\_SOFT != 0.0:  
 soft = df["suggest\_save\_for\_holiday"].astype(int) \* HOLIDAY\_HOLDOUT\_SOFT  
 df["sv\_holiday\_holdout\_soft"] = soft  
 df["spot\_value\_score"] = (df["spot\_value\_score"] + soft).clip(0,1)  
 return df  
  
  
def add\_buckets(df: pd.DataFrame) -> pd.DataFrame:  
 def bucket\_row(r):  
 x = r["spot\_value\_score"]  
 if x >= HI\_THRESH and r.get("week", 99) <= 6: return "Medium" # early-week demotion  
 if x >= HI\_THRESH: return "High"  
 if x >= MED\_THRESH: return "Medium"  
 return "Low"  
 df["spot\_value"] = df.apply(bucket\_row, axis=1)  
 return df  
  
  
def main():  
 ap = argparse.ArgumentParser(); ap.add\_argument("--week", type=str, default="all"); ap.add\_argument("--dry-run", action="store\_true"); ap.add\_argument("--out", type=Path, default=None)  
 args = ap.parse\_args()  
  
 if not ROADMAP.exists():  
 raise FileNotFoundError(ROADMAP)  
  
 base = pd.read\_csv(ROADMAP)  
 base = base.loc[:, ~base.columns.duplicated(keep="first")]  
  
 df = ensure\_columns(base.copy())  
 df = base\_score(df)  
 df = add\_rating\_component(df)  
 df = add\_dvoa\_component(df) # LIVE DVOA level + trend + band  
 df = add\_injury\_component(df)  
 df = add\_holiday\_penalty(df)  
 df = add\_future\_scarcity\_boost(df) # scarcity + now-or-never  
 df = add\_holiday\_highlights(df) # UI flags (no score effect by default)  
 df = add\_buckets(df) # fixed thresholds  
  
 weeks = \_parse\_weeks(args.week)  
 if weeks is not None:  
 mask = df["week"].isin(weeks)  
 out = base.copy()  
 for col in df.columns:  
 if col in out.columns:  
 out.loc[mask, col] = df.loc[mask, col]  
 else:  
 out[col] = pd.NA; out.loc[mask, col] = df.loc[mask, col]  
 else:  
 out = df  
  
 target = args.out or ROADMAP  
 out.to\_csv(target, index=False)  
 print("✅ Saved:", target)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

# B) Locked spec — Roadmap + Score (current state)

## Inputs & files

* **FTN weekly CSV** → python -m scripts.ingest\_ftn\_dvoa\_snapshot <csv> →
  + data/dvoa/dvoa\_timeseries\_2025.csv (all columns + SEASON/SNAPSHOT\_DATE/WEEK, parsed TOT\_DVOA\_PCT)
  + data/dvoa/dvoa\_weekly\_latest.csv (32 rows, per team)
* **Roadmap**: picks/survivor/survivor\_roadmap\_expanded.csv
* **Trends writer**: python -m scripts.compute\_dvoa\_trends → writes (LIVE only):
  + team\_tot\_dvoa\_pp, opp\_tot\_dvoa\_pp, dvoa\_gap\_pp, dvoa\_gap\_dec, trend3\_pp, trend\_band

## Scoring components

Let score = 0.

1. **Win / Home / Rest**
   * sv\_win = W\_WIN \* logistic(projected\_win\_prob) with k=6
   * sv\_home = W\_HOME \* 1{home}
   * sv\_rest = W\_REST \* scale(rest\_days, 4..10)
2. **Rating gap**: sv\_rating = W\_RATING \* tanh(rating\_gap / RATING\_WIDTH)
3. **DVOA (LIVE)**
   * Level: level = clip(dvoa\_gap\_dec, ±LEVEL\_CAP); sv\_dvoa\_level = W\_DVOA\_LEVEL \* level
   * Trend: trend\_norm = clip(trend3\_pp / 10pp, ±1); sv\_dvoa\_trend = clip(W\_DVOA\_TREND \* trend\_norm, ±MAX\_TREND\_BONUS); damp ×0.4 if Week < 5
   * Band: map {Up:+0.015, Down:-0.012}
   * sv\_dvoa = level + trend + band
4. **Injury**: sv\_injury = W\_INJURY \* clip(injury\_adjustment, ±INJURY\_CAP)
5. **Holiday penalty** (only if holiday weeks)
6. **Scarcity & Now-or-Never** (per team sequence)
   * max\_future\_prob from next game onward
   * sv\_scarcity = W\_SCARCITY\_TOTAL \* clip((wp - max\_future) / 0.30, 0..1)
   * sv\_now\_or\_never = NOW\_NEVER\_BONUS \* clip((wp - max\_future - 0.05) / 0.10, 0..1)
7. **Bucket** (fixed thresholds; early demote):
   * if score >= 0.52 and Week ≤ 6 → **Medium**
   * elif score >= 0.52 → **High**
   * elif score >= 0.42 → **Medium**
   * else → **Low**

All intermediate contributions are written into the roadmap with sv\_\* columns.

# C) Lock & run

**PowerShell**

# 1) Ensure latest LIVE DVOA is written into roadmap  
python -m scripts.ingest\_ftn\_dvoa\_snapshot "C:\Users\Spencer\Downloads\ftn\_dvoa.csv"  
python -m scripts.compute\_dvoa\_trends  
  
# 2) Score with tuned weights (this file)  
python -m scripts.spot\_value\_updates  
  
# 3) Diagnostics  
python -m scripts.spot\_value\_bucket\_counts  
python -m scripts.spot\_value\_score

If Medium/High are still too light, tweak **only**: W\_DVOA\_LEVEL ±0.02, HI\_THRESH ±0.01.

# D) Quick live test flow

1. Open the roadmap and filter to **Week 1**.
2. Sort by spot\_value\_score (desc). Check top 3–5 candidates.
3. Verify flags:
   * is\_now\_or\_never == True for any thin-team spike
   * suggest\_save\_for\_holiday == True appropriately
4. Sanity-scan DVOA columns (team\_tot\_dvoa\_pp, opp\_tot\_dvoa\_pp, dvoa\_gap\_dec, trend3\_pp).

# E) What’s next (Millions)

* Reuse the same ingest/trend pipeline; extend the scorer to output **ATS** and **total**-oriented angles (different weights/targets).
* Add opponent exposure balancing at the slate level.
* Introduce a Monte Carlo layer for **win total** or **cover** probabilities per week.

— End of v1.0 lock —