سینا رضایی | **به نام خدا** | **401173007**

سهم های انتخابی:

ر : رتاپ	ض : نداریم	ا : البرز	ی : نداریم

الف) سبد سرمایه‌گذاری (پرتفوی) خود را از نظر بازده، ریسک و درصد وزنی برای هر سهم مشخص نمایید.

MV (Mean–Variance / Standard Deviation)

بازده

	Stock	StartDate	EndDate	StartPrice	EndPrice	TotalReturn(%)
0	رتاپ	1394-09-18	1404-09-18	2693.0	2889.0	7.28
1	البرز	1394-09-18	1404-09-18	1100.0	2422.0	120.18

Portfolio daily volatility (std): 0.03058

	weights
رتاپ	0.891348
البرز	0.108652

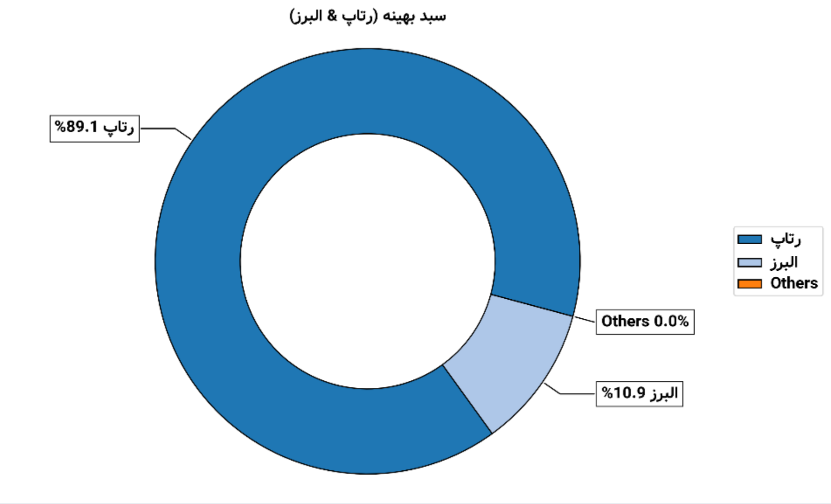
ریسک

```
STD_retap  = np.std(returns["رتاپ"])
STD_alborz = np.std(returns["البرز"])

print(f"Std Dev رتاپ : {STD_retap:.5f}")
print(f"Std Dev البرز : {STD_alborz:.5f}")
```

Std Dev 0.02841 : رتاپ

Std Dev 0.07210 : البرز

درصد وزنی



MAD (Mean Absolute Deviation)

بازده

	Stock	StartDate	EndDate	StartPrice	EndPrice	TotalReturn(%)
0	رتاپ	1394-09-18	1404-09-18	2693.0	2889.0	7.28
1	البرز	1394-09-18	1404-09-18	1100.0	2422.0	120.18

Portfolio daily MAD risk : 0.01747

	weights
رتاپ	0.705676
البرز	0.294324

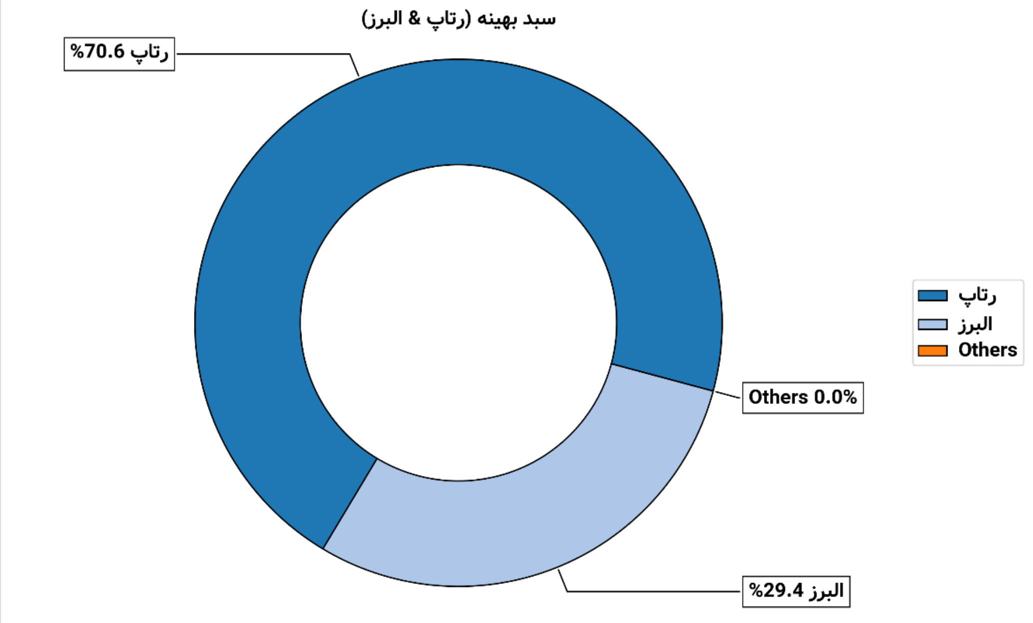
ریسک

```
# MAD for each stock (mean absolute deviation around its own
mean)

retap_ret   = returns["رتاپ"]

alborz_ret  = returns["البرز"]

MAD_retap   = np.mean(np.abs(retap_ret  - retap_ret.mean()))
MAD_alborz  = np.mean(np.abs(alborz_ret - alborz_ret.mean()))

print(f"MAD رتاپ: {MAD_retap:.5f}")

print(f"MAD البرز : {MAD_alborz:.5f}")
```

MAD 0.01864 : رتاپ

MAD 0.02443 : البرز

درصد وزنی



CDaR (Conditional Drawdown at Risk)

بازده

	Stock	StartDate	EndDate	StartPrice	EndPrice	TotalReturn(%)
0	رتاپ	1394-09-18	1404-09-18	2693.0	2889.0	7.28
1	البرز	1394-09-18	1404-09-18	1100.0	2422.0	120.18

Portfolio daily CDaR (alpha=0.95): 0.87374

	weights
رتاپ	0.530334
البرز	0.469666

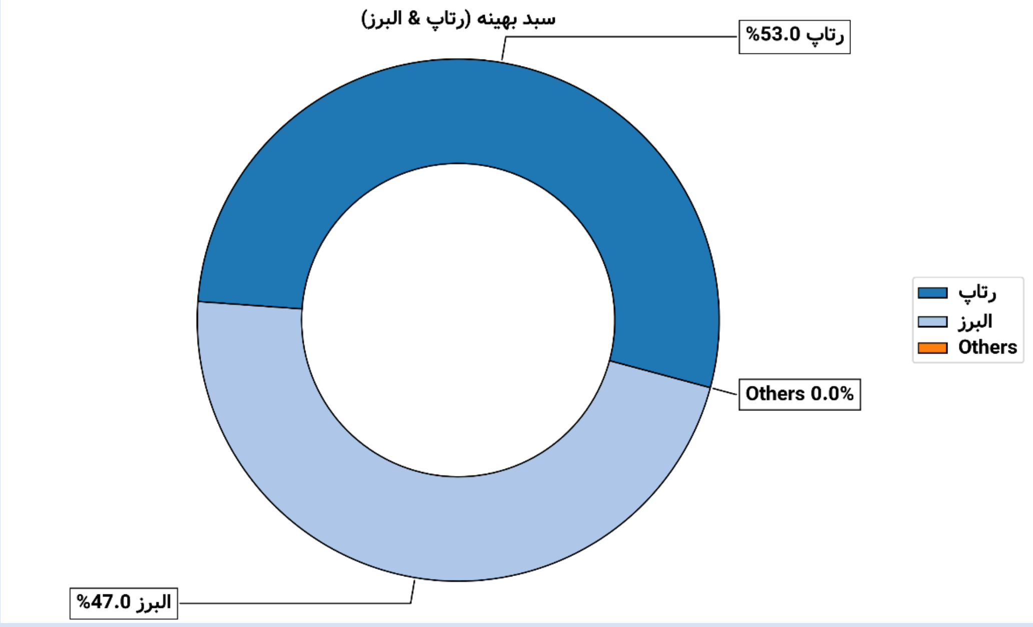
ریسک

```
# Returns for each stock

retap_ret  = returns["رتاپ"]
alborz_ret = returns["البرز"]

alpha = 0.95  # confidence level for CDaR

def calc_cdar(r, alpha=0.95):
    # 1) Cumulative wealth
    cum_wealth = (1 + r).cumprod()
    # 2) Running max of wealth
    running_max = cum_wealth.cummax()
    # 3) Drawdowns (fraction below peak)
    drawdowns = 1 - (cum_wealth / running_max)
    # 4) CDaR = mean of worst (1 - alpha) drawdowns
    threshold_dd = drawdowns.quantile(alpha)
    cdar = float(drawdowns[drawdowns >= threshold_dd].mean())
    return cdar

CDaR_retap  = calc_cdar(retap_ret,  alpha=alpha)
CDaR_alborz = calc_cdar(alborz_ret, alpha=alpha)

print(f"CDaR  رتاپ(alpha={alpha}) : {CDaR_retap:.5f}")
print(f"CDaR البرز (alpha={alpha}) : {CDaR_alborz:.5f}")
```

CDaR رتاپ (alpha=0.95) : 0.91718

CDaR البرز (alpha=0.95) : 0.94092

درصد وزنی



ب) همبستگی بین سهام های انتخابی خود را به دست بیاورید.

همبستگی بین دو سهم

```
# Rename price column to stock names (use "Close" or "Adj Close" depending on df1/df2)
df1_prices = df1.rename(columns={"Close": "رتاپ"})
df2_prices = df2.rename(columns={"Close": "البرز"})

# Keep only the price columns and align on index
merged_df0 = pd.concat([df1_prices["رتاپ"], df2_prices["البرز"]], axis=1)

# Daily simple returns
returns = merged_df0.pct_change().dropna()

# Correlation between the two stocks as a single float
corr_value = returns["رتاپ"].corr(returns["البرز"])  # pandas gives a scalar

print(f"Correlation between رتاپ and البرز: {corr_value:.5f}")
```

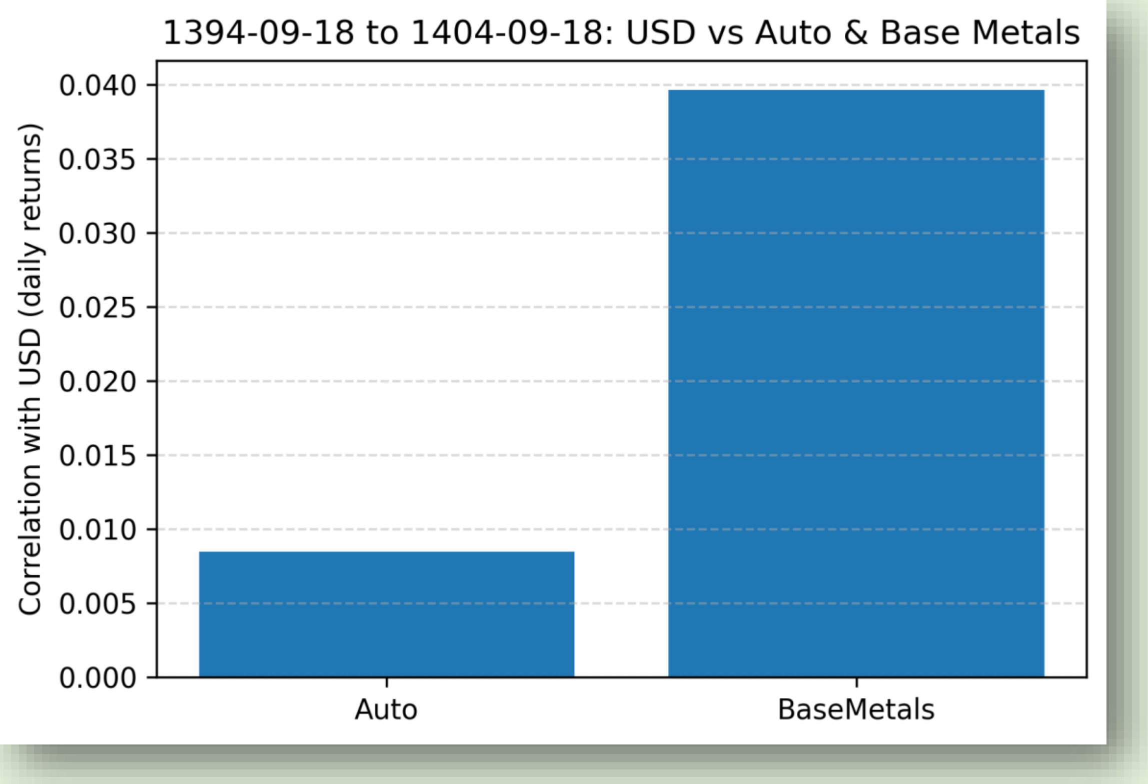
```
Correlation between رتاپ and البرز: 0.09654
```

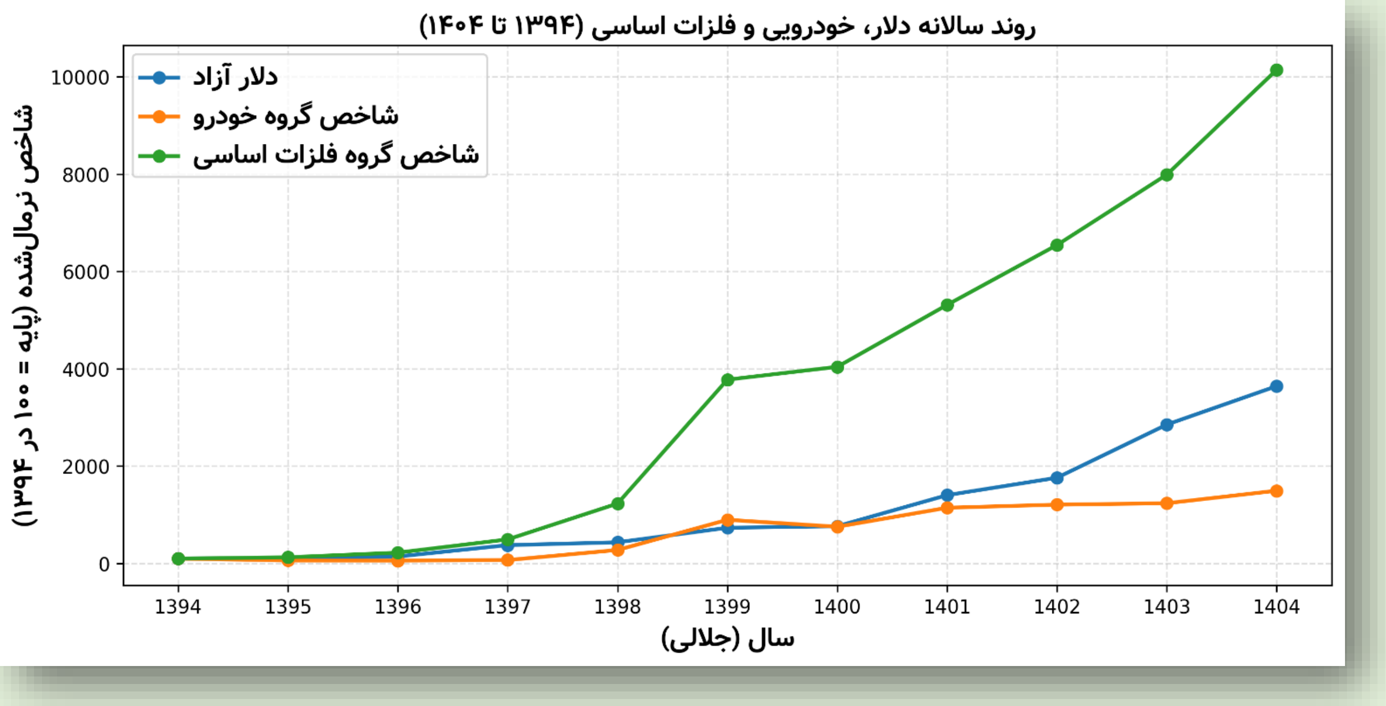
ج) همبستگی بین دلار با گروه فلزات اساسی و گروه خودرو را برای ده سال اخیر به دست آورید و روی نمودار نمایش دهید.

همبستگی بین دلار با گروه فلزات اساسی و خودرو

```
start_date = "1394-09-18"
end_date   = "1404-09-18"

# 1) Download data
usd, auto, base_metals = (
    fpy.Get_USD_RIAL(start_date=start_date, end_date=end_date, ignore_date=False, show_weekday=False, double_date=False),
    fpy.Get_SectorIndex_History(sector="خودرو", start_date=start_date, end_date=end_date, ignore_date=False, just_adj_close=True, show_weekday=False, double_date=False),
    fpy.Get_SectorIndex_History(sector="فلزات اساسی", start_date=start_date, end_date=end_date, ignore_date=False, just_adj_close=True, show_weekday=False, double_date=False))

# ---------- KEY PART: fix index & duplicates ----------
def prepare_df(df):
    df = df.copy()
    # If J-Date column exists, use it as index (common in finpy_tse)
    if "J-Date" in df.columns:
        df = df.set_index("J-Date")
    # Drop duplicate index labels (keep the last observation)
    df = df[~df.index.duplicated(keep="last")]
    # (Optional) sort by index just in case
    df = df.sort_index()
    return df

usd         = prepare_df(usd)
auto        = prepare_df(auto)
base_metals = prepare_df(base_metals)

# ---------- 2) Build a combined price DataFrame ----------
usd_close  = usd["Close"].rename("USD")
auto_close = auto["Adj Close"].rename("Auto")
bm_close   = base_metals["Adj Close"].rename("BaseMetals")

df = pd.concat([usd_close, auto_close, bm_close], axis=1).dropna()

# ---------- 3) Returns & correlations ----------
returns = df.pct_change().dropna()

corr_auto = returns["USD"].corr(returns["Auto"])
corr_bm   = returns["USD"].corr(returns["BaseMetals"])

print(f"Correlation (USD vs Auto):       {corr_auto:.5f}")
print(f"Correlation (USD vs BaseMetals): {corr_bm:.5f}")

# ---------- 4) Simple bar chart ----------
plt.figure(figsize=(6, 4))
plt.bar(["Auto", "BaseMetals"], [corr_auto, corr_bm])
plt.ylabel("Correlation with USD (daily returns)")
plt.title("1394-09-18 to 1404-09-18: USD vs Auto & Base Metals")
plt.grid(axis="y", linestyle="--", alpha=0.5)
plt.show()
```

```
Correlation (USD vs Auto):       0.00845
Correlation (USD vs BaseMetals): 0.03963
```





توضیحاتی درباره هر یک از ریسک های استفاده شده در ارزیابی پرتفولیو

نام ریسک	توضیحات	کد استفاده شده

MV	**What it means**
	- In Riskfolio, `rm = 'MV'` means the risk of the portfolio is measured by **standard deviation of returns**.
	- The optimizer chooses weights w that **minimize the standard deviation** given your other settings (here `obj = 'MinRisk'`).
	Mathematically, if (Σ) is the covariance matrix of returns and (w) is the weight vector:
	$\sigma_{port} = \sqrt{w^T \sum w}$
	So:
	- **Higher MV** → more volatile portfolio (returns move around more).
	- **Lower MV** → smoother, more stable returns.
	In your notebook, this MV measure is used **inside** `port.optimization(...)` to find the “optimal” portfolio weights.
		`model = 'Classic'`
		`rm = 'MV'  # Risk measure used`
		`obj = 'MinRisk'`
		`w = port.optimization(model=model, rm=rm, obj=obj, rf=rf, l=l, hist=hist)`
		And in your “Risk Measures available” comment:
		`# 'MV': Standard Deviation.`

MAD	This is:
	$MAD = \mathbb{E}\lvert R_t - \mu\rvert$
	- (R_t) = daily return
	- (μ) = average daily return
	So, it measures the **average absolute deviation from the mean**, not squared like variance. It is:
	- Easier to interpret: “on average, returns are this far away from their mean”.
	Less sensitive to extreme outliers than variance/standard deviation.
	
	Steps:
	1. Build **portfolio daily returns** from individual stock returns + weights.
	2. Compute the **mean** portfolio return.
	3. Take the **absolute deviation** of each day from that mean and average them → portfolio MAD.
	So, `port_mad` tells you:
	“On an average day, the portfolio return is this far away from its typical (mean) daily return.”
	If your returns are daily, MAD is a **daily risk measure**.
	You could also use MAD **inside** Riskfolio by setting `rm = 'MAD'` in the optimization call, instead of `'MV'`.
		a) MAD for each stock
		`retap_ret   = returns["رتاپ"]`
		`alborz_ret  = returns["البرز"]`
		`MAD_retap   = np.mean(np.abs(retap_ret  - retap_ret.mean()))`
		`MAD_alborz  = np.mean(np.abs(alborz_ret - alborz_ret.mean()))`
		
		`print(f"MAD رتاپ  : {MAD_retap:.5f}")`
		`print(f"MAD البرز : {MAD_alborz:.5f}")`
		
		b) MAD for the **portfolio**
		`weights = w.squeeze()`
		`port_ret_series = returns.mul(weights, axis=1).sum(axis=1)`
		`port_ret = float(port_ret_series.mean())`
		
		`port_mad = float((port_ret_series - port_ret).abs().mean())`
		
		`print(f"Portfolio expected daily return : {port_ret:.5f}")`
		`print(f"Portfolio daily MAD risk        : {port_mad:.5f}")`

CDaR	1. **Cumulative wealth**: simulate investing 1 unit and compounding returns.
	$W_t = \prod_{i=1}^{t}(1 + R_i)$
	2. **Running maximum**: for each day, the highest wealth level up to that day.
	3. **Drawdown series**:
	$DD_t = 1 - \frac{W_t}{max_{s \le t} W_s}$
	- 0 means “at the peak”.
	- 0.2 means “20% below the previous peak”.
	4. **CDaR at level α**: look at the worst $(1 - \alpha)$ tail of drawdowns and take their **average**.
	○ With `alpha = 0.95`, you average the **worst 5% drawdowns**.
	Interpretation:
	- CDaR focuses on **big, persistent losses from peaks** (path-dependent risk).
	- Higher CDaR → the asset tends to experience **deeper or more severe drawdowns**.
		a) CDaR for each stock
		`retap_ret  = returns["رتاپ"]`
		`alborz_ret = returns["البرز"]`
		
		`alpha = 0.95  # confidence level for CDaR`
		
		`def calc_cdar(r, alpha=0.95):`
		`    # 1) Cumulative wealth`
		`    cum_wealth = (1 + r).cumprod()`
		`    # 2) Running max of wealth`
		`    running_max = cum_wealth.cummax()`
		`    # 3) Drawdowns (fraction below peak)`
		`    drawdowns = 1 - (cum_wealth / running_max)`
		`    # 4) CDaR = mean of worst (1 - alpha) drawdowns`
		`    threshold_dd = drawdowns.quantile(alpha)`
		`    cdar = float(drawdowns[drawdowns >=`
		`threshold_dd].mean())`
		`    return cdar`
		
		`CDaR_retap  = calc_cdar(retap_ret,  alpha=alpha)`
		`CDaR_alborz = calc_cdar(alborz_ret, alpha=alpha)`
		
		b) CDaR for the portfolio
		Later you do the same on the **portfolio**:
		`weights = w.squeeze()`
		`port_ret_series = returns.mul(weights, axis=1).sum(axis=1)`
		`port_ret = float(port_ret_series.mean())`
		
		`alpha = 0.95  # confidence level for CDaR`
		
		`cum_wealth = (1 + port_ret_series).cumprod()`
		`running_max = cum_wealth.cummax()`
		`drawdowns = 1 - (cum_wealth / running_max)`
		
		`threshold_dd = drawdowns.quantile(alpha)`
		`cdar = float(drawdowns[drawdowns >= threshold_dd].mean())`
		
		`print(f"Portfolio expected daily return     :`
		`{port_ret:.5f}")`
		`print(f"Portfolio daily CDaR (alpha={alpha}): {cdar:.5f}")`