***

I employed SAS as the data cleaning software. The data merging process proved highly complex. Initially, I utilised Bao et al. (2021), named data_FraudDetection_JAR2020. This dataset encompasses AAER's fraudulent company data from 1991 to 2014, alongside 28 original financial data points and 14 financial ratio data points utilised in their research. I incorporated market, audit, and governance data sourced from three distinct WRDS databases: CRSP, Boardex, and Audit Analytics. These databases employ distinct identifiers for company recognition, necessitating the identification of corresponding links from the Company Database Linking Matrix to merge datasets. Identifiers such as GVKEY, CIK, companyid, and PERMNO were utilised, resulting in substantial data loss. Firstly, as Audit Analytics' audit fee data commenced in 2000, all data had to be restricted to the 2000-2014 period. Within market data, volatility, adjusted returns, and bid-ask spreads required annual calculations derived from daily data. Regarding governance data, my initial attempt to quantify board independence encountered numerous instances of independent directors leaving their posts in the organisational data. This led to situations where the number of independent directors exceeded the total board membership. Ultimately, I opted to incorporate gender ratios, a measure also employed in other literature. The original data_FraudDetection_JAR2020 dataset contained 146,045 annual observations. After restricting the years to 2000–2014, it comprised 87,411 data points. Following multiple merges and missing value removal, 32,731 annual data points were generated.

Here is the complete code for my data cleaning process.***

```
/* Assign library for CRSP data */

libname crspdata 'C:\Users\Shi\raw data';


/* Check dataset structure */

proc contents data=crspdata.CRSP_DAILY;

run;


/* Step 1: Check data structure and variable names */

proc contents data=crspdata.CRSP_DAILY;

      title "CRSP_DAILY Dataset Structure";

run;
```

```
/* View first 10 observations to confirm variable names */
proc print data=crspdata.CRSP_DAILY (obs=10);
    title "First 10 Observations of CRSP_DAILY";
run;

/* Data preparation and sorting */
proc sort data=crspdata.CRSP_DAILY out=crsp_sorted;
    by PERMNO date;   /* Changed to PERMNO */
    where ret is not null;
run;

/* 1. Calculate daily bid-ask spread */
proc sql;
    create table daily_spread as
    select
        PERMNO,   /* Changed to PERMNO */
        date,
        year(date) as year,
        /* Calculate daily relative bid-ask spread */
        case when ask > 0 and bid > 0 and ask > bid
            then (ask - bid) / ((ask + bid) / 2)
            else . end as daily_spread format=percent8.4,
        ask,
        bid,
        (ask - bid) as absolute_spread   /* Absolute spread */
    from crsp_sorted
    where ask > 0 and bid > 0 and ask > bid;
quit;
```

```
/* Calculate annual average bid-ask spread */
proc sql;
    create table annual_avg_spread as
    select
        PERMNO,    /* Changed to PERMNO */
        year,
        mean(daily_spread) as annual_avg_spread format=percent8.4,
        count(*) as trading_days,
        min(date) as first_date,
        max(date) as last_date
    from daily_spread
    group by PERMNO, year    /* Changed to PERMNO */
    having count(*) >= 120    /* At least 120 trading days */
    order by PERMNO, year;    /* Changed to PERMNO */
quit;

/* Calculate lagged annual spread */
data annual_spread_lag1;
    set annual_avg_spread;
    by PERMNO year;    /* Changed to PERMNO */

    /* Get lagged annual spread */
    lag1_annual_spread = lag(annual_avg_spread);
    lag1_year = lag(year);

    /* Ensure same stock */
    if first.PERMNO then do;    /* Changed to PERMNO */
        lag1_annual_spread = .;
        lag1_year = .;
```

```
    end;

    /* Keep only observations with lagged data */
    if not missing(lag1_annual_spread) and year = lag1_year + 1;

    keep PERMNO year annual_avg_spread lag1_annual_spread lag1_year
trading_days;   /* Changed to PERMNO */
    format annual_avg_spread lag1_annual_spread percent8.4;
run;

/* Calculate daily market-adjusted returns */
proc sql;
    create table daily_adjusted_returns as
    select
        PERMNO,   /* Changed to PERMNO */
        date,
        year(date) as year,

        /* Stock daily return (including delisting returns) */
        coalesce(ret, 0) + coalesce(dlret, 0) as daily_stock_return,

        /* Market daily return */
        vwretd as daily_market_return,

        /* Daily market-adjusted return */
        (1 + coalesce(ret, 0) + coalesce(dlret, 0)) / (1 + vwretd) - 1 as
daily_adjusted_return,

        /* Raw return for volatility calculation */
        coalesce(ret, 0) as daily_raw_return
```

```
    from crspdata.CRSP_DAILY
    where ret is not null or dlret is not null;
quit;

/* Calculate annual market-adjusted returns */
proc sql;
    create table annual_adjusted_returns as
    select
        PERMNO,    /* Changed to PERMNO */
        year,

        /* Annual market-adjusted return - using geometric mean */
        exp(sum(log(1 + daily_adjusted_return))) - 1 as annual_adjusted_return
format=percent8.2,

        /* Supplementary statistics */
        count(*) as trading_days,
        mean(daily_adjusted_return) as avg_daily_adjusted_return
format=percent8.4,
        std(daily_adjusted_return) as std_daily_adjusted_return
format=percent8.4,
        min(daily_adjusted_return) as min_daily_adjusted_return
format=percent8.4,
        max(daily_adjusted_return) as max_daily_adjusted_return
format=percent8.4

    from daily_adjusted_returns
    where not missing(daily_adjusted_return)
    group by PERMNO, year    /* Changed to PERMNO */
    having count(*) >= 120    /* At least 120 trading days */
```

```
    order by PERMNO, year;    /* Changed to PERMNO */
quit;

/* Get lagged annual market-adjusted returns */
data annual_adjusted_returns_lag;
    set annual_adjusted_returns;
    by PERMNO year;    /* Changed to PERMNO */

    /* Get lagged annual adjusted returns */
    lag1_annual_adjusted_return = lag(annual_adjusted_return);
    lag1_year = lag(year);
    lag1_trading_days = lag(trading_days);

    /* Ensure same stock */
    if first.PERMNO then do;    /* Changed to PERMNO */
        lag1_annual_adjusted_return = .;
        lag1_year = .;
        lag1_trading_days = .;
    end;

    /* Keep only observations with lagged data */
    if not missing(lag1_annual_adjusted_return) and year = lag1_year + 1;

    keep PERMNO year annual_adjusted_return lag1_annual_adjusted_return    /*
Changed to PERMNO */
        trading_days lag1_trading_days lag1_year
        avg_daily_adjusted_return std_daily_adjusted_return;

    format annual_adjusted_return lag1_annual_adjusted_return percent8.2
           avg_daily_adjusted_return std_daily_adjusted_return percent8.4;
```

```
run;

/* Calculate annualized volatility - standard method */
proc sql;
    create table annual_volatility as
    select
        PERMNO,   /* Changed to PERMNO */
        year(date) as year,

        /* Annualized volatility = daily return std dev × √252 */
        std(ret) * sqrt(252) as annual_volatility format=percent8.2,

        /* Supplementary statistics */
        count(*) as trading_days,
        mean(ret) as avg_daily_return format=percent8.4,
        min(ret) as min_daily_return format=percent8.4,
        max(ret) as max_daily_return format=percent8.4,
        std(ret) as daily_volatility format=percent8.4

    from crsp_sorted
    where ret is not null
    group by PERMNO, calculated year   /* Changed to PERMNO */
    having count(*) >= 120   /* At least 120 trading days */
    order by permno, year;   /* Changed to PERMNO */
quit;

/* Ensure annual volatility data is properly sorted */
proc sort data=annual_volatility;
    by PERMNO year;   /* Changed to PERMNO */
```

```
run;

data lag_volatility_data;
    set annual_volatility;
    by PERMNO;   /* Changed to PERMNO */

    /* Set volatility to previous year's value */
    annual_volatility = lag(annual_volatility);

    /* Remove first observation (no previous year data) */
    if not first.PERMNO;   /* Changed to PERMNO */

    format annual_volatility percent8.2;
run;

proc print data=lag_volatility_data (obs=20);
    title "Data Using Previous Year Volatility";
run;

/* Export market data */
proc sql;
    create table CRSP_data_clean as
    select
        s.PERMNO,   /* Changed to PERMNO */
        s.year,

        /* Bid-ask spread data */
        s.annual_avg_spread,
        s.lag1_annual_spread,
```

```
        s.trading_days as spread_days,

        /* Current year return data */
        r.annual_adjusted_return,
        r.trading_days as return_days,

        /* Lagged return data */
        r_lag.annual_adjusted_return as lag1_annual_adjusted_return,
        r_lag.trading_days as lag1_return_days,
        r_lag.year as reference_year,    /* Actual year for lagged data */

        /* Volatility data */
        v.annual_volatility,
        v.trading_days as vol_days

    from annual_spread_lag1 s
    left join annual_adjusted_returns r
        on s.PERMNO = r.PERMNO and s.year = r.year    /* Changed to PERMNO
*/
    left join annual_adjusted_returns r_lag
        on s.PERMNO = r_lag.PERMNO and s.year = r_lag.year + 1    /* Changed
to PERMNO, lagged by one year */
    left join lag_volatility_data v
        on s.PERMNO = v.PERMNO and s.year = v.year    /* Changed to PERMNO
*/
    order by s.PERMNO, s.year;    /* Changed to PERMNO */
quit;

/* Export to CSV file */
proc export data=CRSP_data_clean
```

```
    outfile="C:\Users\Shi\raw data\CRSP_YEAR_CLEAN.csv"
    dbms=csv replace;
run;

/* Process and export audit data */

/* Import data */
proc import datafile="C:\Users\Shi\raw data\audit_fee.csv"
    out=audit_fee
    dbms=csv replace;
    guessingrows=1000;
run;

/* Sort by FISCAL_YEAR */
proc sort data=audit_fee out=audit_fee_sorted;
    by FISCAL_YEAR;
run;

/* Calculate percentile ranks */
proc rank data=audit_fee_sorted out=ranked groups=100;
    by FISCAL_YEAR;
    var AUDIT_FEES NON_AUDIT_FEES TOTAL_FEES;
    ranks audit_rank non_audit_rank total_rank;
run;

/* Calculate required metrics */
data audit_fee_final;
    set ranked;
```

```
/* Create new variables */
cik = COMPANY_FKEY;
fyear = FISCAL_YEAR;

/* Calculate logarithms - handle zero and negative values safely */
if AUDIT_FEES > 0 then ln_audit_fees = log(AUDIT_FEES);
else if AUDIT_FEES = 0 then ln_audit_fees = 0;   /* Set 0 values to 0 */
else ln_audit_fees = .;   /* Set negative values to missing */

if NON_AUDIT_FEES > 0 then ln_non_audit_fees = log(NON_AUDIT_FEES);
else if NON_AUDIT_FEES = 0 then ln_non_audit_fees = 0;
else ln_non_audit_fees = .;

if TOTAL_FEES > 0 then ln_total_fees = log(TOTAL_FEES);
else if TOTAL_FEES = 0 then ln_total_fees = .;
else ln_total_fees = .;

/* Calculate ratios - handle division by zero safely */
if TOTAL_FEES > 0 then do;
    audit_ratio = AUDIT_FEES / TOTAL_FEES;
    non_audit_ratio = NON_AUDIT_FEES / TOTAL_FEES;
end;
else do;
    audit_ratio = .;
    non_audit_ratio = .;
end;

/* Convert ranks to percentile proportions (0-1 range) */
audit_fees_pctrank = audit_rank / 99;        /* 0-1 range */
```

```
    non_audit_fees_pctrank = non_audit_rank / 99;
    total_fees_pctrank = total_rank / 99;



    /* Formatting */
    format audit_fees_pctrank non_audit_fees_pctrank total_fees_pctrank
percent8.1;
    format audit_ratio non_audit_ratio percent8.2;

    keep cik fyear AUDIT_FEES NON_AUDIT_FEES TOTAL_FEES
        ln_audit_fees ln_non_audit_fees ln_total_fees
        audit_ratio non_audit_ratio
        audit_fees_pctrank non_audit_fees_pctrank total_fees_pctrank;
run;



/* View first 20 rows of data */
proc print data=audit_fee_final (obs=20);
    title "First 20 Rows of Processed Data";
run;

/* Export to CSV */
proc export data=audit_fee_final
    outfile="C:\Users\Shi\raw data\Audit_Fee_clean.csv"
    dbms=csv replace;
run;

/* Import board data */
libname board "C:\Users\Shi\raw data\";
```

```
/* Check data structure and rowtype distribution */
proc freq data=board.board;
    tables rowtype / missing;
    title "rowtype Field Distribution";
run;

/* Filter for rowtype = board member */
data board_members;
    set board.board;
    where upcase(strip(rowtype)) = 'BOARD MEMBER';
run;

/* Check filtering results */
proc means data=board_members n;
    title "Number of Observations After Filtering";
run;

proc contents data=board.board varnum;
run;

/* First sort by companyid and annualreportdate */
proc sort data=board.board out=sorted_board;
    by companyid annualreportdate;
run;

proc sql;
    create table role_summary as
    select
        distinct strip(upcase(rolename)) as role_name
```

```
    from board.board
    where not missing(rolename)
    order by calculated role_name;
quit;

proc print data=role_summary;
    title "All Different Role Names";
run;

/* Calculate chairman-CEO duality and include required variables */
proc sql;
    create table chairman_ceo_duality as
    select
        companyid,
        year(annualreportdate) as fyear,

        /* Check if chairman and CEO roles are combined */
        max(case
            when index(upcase(rolename), 'CHAIRMAN') > 0 and
                 index(upcase(rolename), 'CEO') > 0
            then 1
            else 0
        end) as chairman_ceo_duality,

        /* Include required keep variables */
        max(genderratio) as genderratio,
        max(numberdirectors) as numberdirectors,

        /* Optional: count sample size */
```

```
        count(distinct directorid) as total_directors,
        count(*) as total_records

    from board.board
    group by companyid, calculated fyear
    order by companyid, calculated fyear;
quit;

/* Keep only required variables */
data chairman_ceo_final;
    set chairman_ceo_duality;
    keep companyid fyear chairman_ceo_duality genderratio numberdirectors;
run;

proc print data=chairman_ceo_final (obs=20);
    title "First 20 Rows of Processed Board Data";
run;

/* Save as CSV file */
proc export data=chairman_ceo_final
    outfile="C:\Users\Shi\raw data\governance_metrics.csv"
    dbms=csv
    replace;
run;



/* Merge final dataset */
/* Import all datasets - use consistent lowercase naming */
proc import datafile="data_FraudDetection_JAR2020.csv" out=main_data
dbms=csv replace;
```

```
    getnames=yes;
    guessingrows=1000;
run;

proc import datafile="codelink.csv" out=codelink dbms=csv replace;
    getnames=yes;
    guessingrows=1000;
run;

proc import datafile="crsp_comp.csv" out=crsp_comp dbms=csv replace;
    getnames=yes;
    guessingrows=1000;
run;

proc import datafile="Audit_Fee_clean.csv" out=audit_fee dbms=csv replace;
    getnames=yes;
    guessingrows=1000;
run;

proc import datafile="CRSP_YEAR_CLEAN.csv" out=crsp_year dbms=csv replace;
    getnames=yes;
    guessingrows=1000;
run;

proc import datafile="governance_metrics.csv" out=governance dbms=csv
replace;
    getnames=yes;
    guessingrows=1000;
run;
```

```
/* Step 2: Process main dataset, keep data from 2000-2014 */
data main_data_filtered;
    set main_data;
    where fyear >= 2000 and fyear <= 2014;
run;

/* 1. Ensure both datasets are sorted by gvkey and fyear */
proc sort data=main_data_filtered;
    by gvkey fyear;
run;

proc sort data=crsp_comp;
    by gvkey fyear;
run;

proc sql;
    create table main_data_with_crsp as
    select
        a.*,                /* All variables from main dataset data_FraudDetection_JAR2020 */
        b.LPERMNO,          /* Add LPERMNO from crsp_comp */
        b.LPERMCO,          /* Add LPERMCO from crsp_comp */
        b.CIK               /* Add CIK from crsp_comp */
    from main_data_filtered a
    left join crsp_comp b
        on a.gvkey = b.gvkey and a.fyear = b.fyear;
quit;

/* Check merge results */
proc means data=main_data_with_crsp n nmiss;
```

```
    var LPERMNO LPERMCO;
    title "Missing Values After Merging Main Dataset with CRSP";
run;

/* View merged data */
proc print data=main_data_with_crsp (obs=20);
    var gvkey fyear LPERMNO LPERMCO;
    title "First 20 Observations After Merging Main Dataset with CRSP";
run;

/* Calculate match success rate */
proc sql;
    select
        count(*) as total_observations,
        sum(case when LPERMNO is not null then 1 else 0 end) as
matched_permno,
        sum(case when LPERMCO is not null then 1 else 0 end) as
matched_permco,
        calculated matched_permno / calculated total_observations as
permno_match_rate format=percent8.2,
        calculated matched_permco / calculated total_observations as
permco_match_rate format=percent8.2
    from main_data_with_crsp;
quit;

/* Perform codelink merge */
proc sql;
    create table main_data_with_codelink as
    select
        a.*,                    /* All variables from main dataset */
        b.companyid,            /* Add companyid from codelink */
```

```
        b.PERMCO                /* Add PERMCO from codelink */
    from main_data_with_crsp a
    left join codelink b
        on a.gvkey = b.gvkey;
quit;

/* Check merge results */
proc means data=main_data_with_codelink n nmiss;
    var companyid PERMCO;
    title "Missing Values for companyid and PERMCO After Merge";
run;

/* View merged data */
proc print data=main_data_with_codelink (obs=20);
    var gvkey fyear companyid PERMCO LPERMNO LPERMCO;
    title "First 20 Observations After Merging with Codelink";
run;

/* Calculate match success rate */
proc sql;
    select
        count(*) as total_observations,
        sum(case when companyid is not null then 1 else 0 end) as
matched_companyid,
        sum(case when PERMCO is not null then 1 else 0 end) as
matched_PERMCO,
        calculated matched_companyid / calculated total_observations as
companyid_match_rate format=percent8.2,
        calculated matched_PERMCO / calculated total_observations as
PERMCO_match_rate format=percent8.2
    from main_data_with_codelink;
```

```
quit;

/* First check if main dataset has necessary matching variables */
proc contents data=main_data_with_codelink;
    title "Variable List of Main Dataset";
run;

proc contents data=audit_fee;
    title "Variable List of Audit Fee Dataset";
run;

proc contents data=crsp_year;
    title "Variable List of CRSP Annual Dataset";
run;

proc contents data=governance;
    title "Variable List of Corporate Governance Dataset";
run;

/* Final merge of all datasets */
proc sql;
    create table FINAL_MERGED_DATASET as
    select
        /* All variables from main dataset */
        main.*,

        /* Merge audit fee related variables from AUDIT_FEE */
        audit.ln_audit_fees,
        audit.ln_non_audit_fees,
```

```
    audit.ln_total_fees,
    audit.audit_ratio,
    audit.non_audit_ratio,
    audit.audit_fees_pctrank,
    audit.non_audit_fees_pctrank,
    audit.total_fees_pctrank,

    /* Merge market data variables from CRSP_YEAR */
    crspyr.lag1_annual_spread,
    crspyr.lag1_annual_adjusted_return,
    crspyr.annual_volatility,

    /* Merge corporate governance variables from GOVERNANCE */
    gov.chairman_ceo_duality,
    gov.genderratio,
    gov.numberdirectors

from MAIN_DATA_WITH_CODELINK as main

/* Merge audit fee data */
left join AUDIT_FEE as audit
    on main.cik = audit.cik and main.fyear = audit.fyear

/* Merge market data - treat year in CRSP_YEAR as fyear */
left join CRSP_YEAR as crspyr
    on main.LPERMNO = crspyr.PERMNO and main.fyear = crspyr.year

/* Merge corporate governance data */
left join GOVERNANCE as gov
```

```
        on main.companyid = gov.companyid and main.fyear = gov.fyear

    order by main.gvkey, main.fyear;
quit;


/* Check final merge results */
proc contents data=final_merged_dataset;
    title "Structure of Final Merged Dataset";
run;

proc means data=final_merged_dataset n nmiss;
    var ln_audit_fees ln_non_audit_fees ln_total_fees
        lag1_annual_spread lag1_annual_adjusted_return annual_volatility
        chairman_ceo_duality genderratio numberdirectors;
    title "Missing Values in Final Dataset";
run;

/* View only first 10 observations to avoid excessive output */
proc print data=final_merged_dataset (obs=100);
    title "First 10 Observations of Final Dataset - All Variables";
run;

data final_merged_dataset_clean;
    set final_merged_dataset;
    where not missing(LPERMNO) and not missing(LPERMCO);
run;

/* More concise method, list all variables to check */
```

```
proc sql;
    create table final_merged_dataset_clean as
    select *
    from final_merged_dataset
    where LPERMNO is not null and
          LPERMCO is not null and
          ln_audit_fees is not null and
          ln_non_audit_fees is not null and
          ln_total_fees is not null and
          lag1_annual_spread is not null and
          lag1_annual_adjusted_return is not null and
          annual_volatility is not null and
          chairman_ceo_duality is not null and
          genderratio is not null and
          numberdirectors is not null;
quit;

/* Compare observations before and after processing */
proc sql;
    title "Comparison Before and After Missing Value Treatment";
    select
        (select count(*) from final_merged_dataset) as original_obs,
        (select count(*) from final_merged_dataset_clean) as cleaned_obs,
        calculated original_obs - calculated cleaned_obs as deleted_obs
    from final_merged_dataset(obs=1);
quit;

/* Save final dataset */
data final_analysis_data;
```

```
    set final_merged_dataset_clean;
run;

/* Export as CSV file */
proc export data=final_merged_dataset_clean
    outfile="final_merged_dataset.csv"
    dbms=csv replace;
run;

/* Step 5: View basic information of final dataset */
proc contents data=combin_data;
run;

proc print data=combin_data(obs=10);
run;

/* Step 6: Check data quality */
proc means data=combin_data n nmiss;
    var ln_audit_fees annual_adjusted_return total_directors;
run;

proc freq data=combin_data;
    tables fyear;
run;

/* Save final dataset */
data work.combin_data_final;
    set combin_data;
run;
```

```
/* View basic information of final dataset */
proc contents data=combin_data;
run;

proc print data=combin_data(obs=10);
run;
```