Assignment 1

Balaaditya Matcha

December 16, 2025

Contents

1 Question 1: Linear Regression

A linear regression model with $\surd$ predictors just means it has $\surd$ inputs to it, i.e, $x_{i1}, x_{2i}, \ldots, x_{ip}$.
Let us use $y = [y_1\ y_2\ \ldots\ y_n]$ as the vector with it's elements being **response variables** in the following subsections along with β, X.

1.1 Part 1

The equation of the multiple linear regression model is

$$y = X\beta + \epsilon$$

where y, X, β are as defined above and ϵ representing the vector with it's elements(ϵ_i) representing the deviations of model's output wrt observed responses.

Variables:

- n : Number of samples
- p : Number of predictors

- y_i : Response variable
- x_{ij} : j^{th} predictor of i^{th} sample

Parameters:

- β_0 : The intercept of the modelled line.
- β_i : The slope/weight wrt each input(x_{ij})
- ϵ_i : Error/Deviation(already explained above)

Assumptions:

1.1.1 Linearity:

The relationship between the predictors and the response is linear in parameters:

$$\mathbb{E}[y \mid X] = X\beta.$$

This is to ensurethe model is correctly specified and it allows OLS to estimate coefficients meaningfully.

1.1.2 Independence of errors:

The error terms are independent:

$$\varepsilon_i \text{ is independent of } \varepsilon_j \quad \text{for } i \neq j.$$

This prevents information leakage across observations and ensures correct variance estimation.

1.1.3 Mean of errors set to 0:

It ensures OLS estimator to be unbiased. If violated, coefficients are systematically wrong.

1.1.4 Homoscedasticity:

$$\text{Var}(\varepsilon_i \mid X) = \sigma^2 \quad \text{for all } i.$$

This ensures standard errors are correct. Without it, inferences becomes unreliable.

1.1.5 No perfect multicollinearity

That is, the columns of X are linearly independent.
This ensures X^TX is invertible. **Without it, OLS estimates do not exist.**

1.1.6 Normality of Errors

This assumption ensures the reliability on hypothesis testings and confidence intervals of the model.

1.2 Part 2

Ordinary least squares (OLS) minimizes the **Sum of Squared Residuals (SSR)**. We can understand that residual means the error/deviation.

SSR: $\sum_{i=1}^{n}(y_i - \sum_{j=0}^{p} x_{ij}\beta_j)^2 = \sum_{i=1}^{n} \epsilon_i^2$
Let us denote the full mean-squared error(MSE) by $\mathcal{J}(\beta)$.

$$\mathcal{J}(\beta) = \frac{1}{n}\|X\beta - y\|^2$$

1.3 Part 3

To get $\hat{\beta}$(the estimated parameters at the end of process), we differentiate the MSE function wrt β and set it to 0.

$$n\mathcal{J}(\beta) = (y - X\beta)^T(y - X\beta)$$

Expanding the quadratic form:

$$n\mathcal{J}(\boldsymbol{\beta}) = (\mathbf{y}^T - \boldsymbol{\beta}^T\mathbf{X}^T)(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$
$$n\mathcal{J}(\boldsymbol{\beta}) = \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\boldsymbol{\beta} - \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{y} + \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta}$$

Since $\mathbf{y}^T\mathbf{X}\boldsymbol{\beta}$ is a scalar, it equals its transpose $\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{y}$. We combine them:

$$n\mathcal{J}(\boldsymbol{\beta}) = \mathbf{y}^T\mathbf{y} - 2\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{y} + \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta}$$

Differentiation:

$$\nabla_{\boldsymbol{\beta}} n\mathcal{J} = \frac{\partial}{\partial \boldsymbol{\beta}}(\mathbf{y}^T\mathbf{y} - 2\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{y} + \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta})$$
$$0 = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\boldsymbol{\beta}$$

Rearranging to solve for $\boldsymbol{\beta}$:

$$\begin{aligned} 2\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} &= 2\mathbf{X}^T\mathbf{y} \\ \mathbf{X}^T\mathbf{X}\boldsymbol{\beta} &= \mathbf{X}^T\mathbf{y} \\ \hat{\boldsymbol{\beta}} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \end{aligned}$$

$$\boxed{\hat{\beta} = (X^T X)^{-1} X^T y}$$

1.4 Part 4

Multicollinearity means one of our inputs(x_{ij}) can be linearly predicted from the others with a **lot of accuracy**. It indicates that our features contain redundant information.
But to fit the question, we will consider that accuracy = 1, i.e, some column in the design matrix x_i is a linear combination of some other columns.
Then, columns of the design matrix are dependent, i.e,

$$\text{rank}(X) < p + 1$$

Since rank($X^T X$) =rank(X), [A standard property]

$$\text{rank}(X^T X) < p + 1$$

and since $X^T X$ is a $(p+1)$x$(p+1)$ matrix, It is a **singular matrix** and it's inverse does not exist.

1.5 Part 5

Since $\hat{y} = X\hat{\beta}$ by definition in linear regression model,

$$\begin{aligned} X^T\hat{y} &= X^T X(X^T X)^{-1} X^T y \\ &= X^T y \end{aligned}$$

$\therefore X^T(y - \hat{y}) = 0$ That is, orthogonal projection of y onto the column space of X is $\hat{y}$.

1.6 Part 6

Given,

$$2n\mathcal{J}(\beta) = (y - X\beta)^T(y - X\beta)$$

Expanding the quadratic form:

$$2n\mathcal{J}(\boldsymbol{\beta}) = (\mathbf{y}^T - \boldsymbol{\beta}^T\mathbf{X}^T)(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$
$$2n\mathcal{J}(\boldsymbol{\beta}) = \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\boldsymbol{\beta} - \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{y} + \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta}$$

Since $\mathbf{y}^T\mathbf{X}\boldsymbol{\beta}$ is a scalar, it equals its transpose $\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{y}$. We combine them:

$$2n\mathcal{J}(\boldsymbol{\beta}) = \mathbf{y}^T\mathbf{y} - 2\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{y} + \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta}$$

Differentiation:

$$\nabla_{\boldsymbol{\beta}} 2n\mathcal{J} = \frac{\partial}{\partial \boldsymbol{\beta}}(\mathbf{y}^T\mathbf{y} - 2\boldsymbol{\beta}^T\mathbf{X}^T\mathbf{y} + \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta})$$

$$\boxed{\nabla_{\boldsymbol{\beta}}\mathcal{J}(\beta) = \frac{1}{n}\mathbf{X}^T(\mathbf{X}\boldsymbol{\beta} - \mathbf{y})}$$

Batch gradient descent update rule:

$$\beta^{(t+1)} = \beta^{(t)} - \mathcal{L}\nabla_\beta J(\beta^{(t)})$$

On substitution of $\nabla_\beta J(\beta^{(t)})$,

$$\beta^{(t+1)} = \beta^{(t)} - \mathcal{L}\frac{1}{n}X^T(X\beta^{(t)} - y)$$

where $\mathcal{L}$ is the learning rate.

1.7 Part 7

Listing 1: Calculation of β_hat and comparisions.

```
# numpy method
pinvX = np.linalg.pinv(X)
beta_np = pinvX @ y

# sklearn method
```

```
X_sklearn = X[:, 1:]
model = LinearRegression(fit_intercept=True)
model.fit(X_sklearn, y)

beta_sklearn = np.concatenate(([model.intercept_], model.coef_))


print("numpy beta: ", beta_np)
print("sklearn beta: ", beta_sklearn)
print("Difference: ", beta_np - beta_sklearn)
```

Both the results are same upto 6 decimals but there are some differences only in the order of 10^{-15}.

1.8 Part 8

Listing 2: Plotting of Residuals vs Fitted values.

```
y_hat = X @ beta_np
res = y - y_hat

plt.scatter(y_hat, res)
plt.xlabel("Fitted values")
plt.ylabel("Residuals")
plt.title("Residuals vs Fitted Values")
plt.show()
```

Homoscedasticity: Equal variance
While the density of points is higher in the center (around 0), the vertical spread of the residuals remains relatively constant across the range of fitted values(roughly between -2 and +2), so the variance is **almost** constant, i.e, satisfying our assumption of **Homoscedasticity**.

1.9 Part 9

Listing 3: Plotting Q-Q Plot of Residuals.

```
stats.probplot(res, dist="norm", plot=plt)
plt.title("Q-Q Plot of Residuals")
```

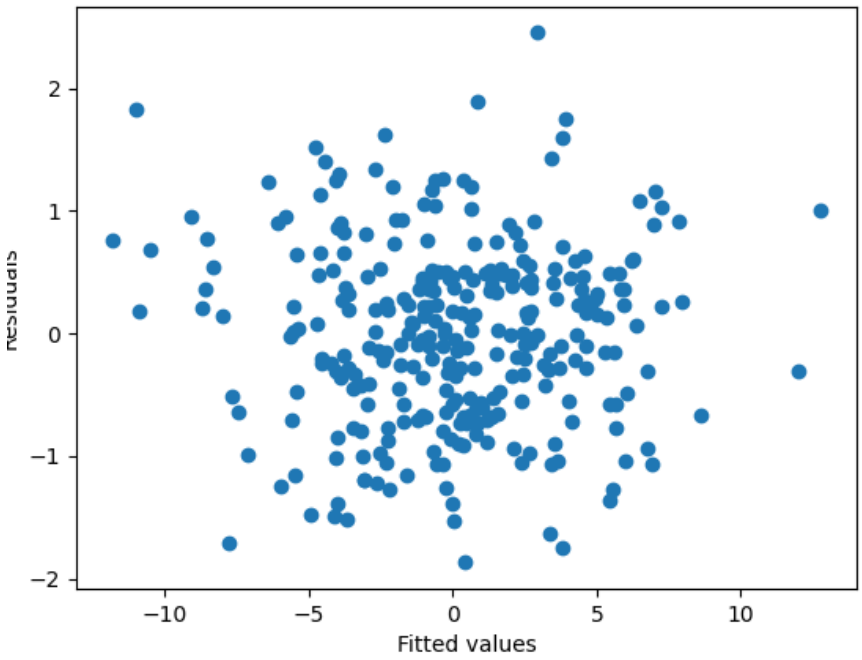


Figure 1: Residuals vs Fitted Values

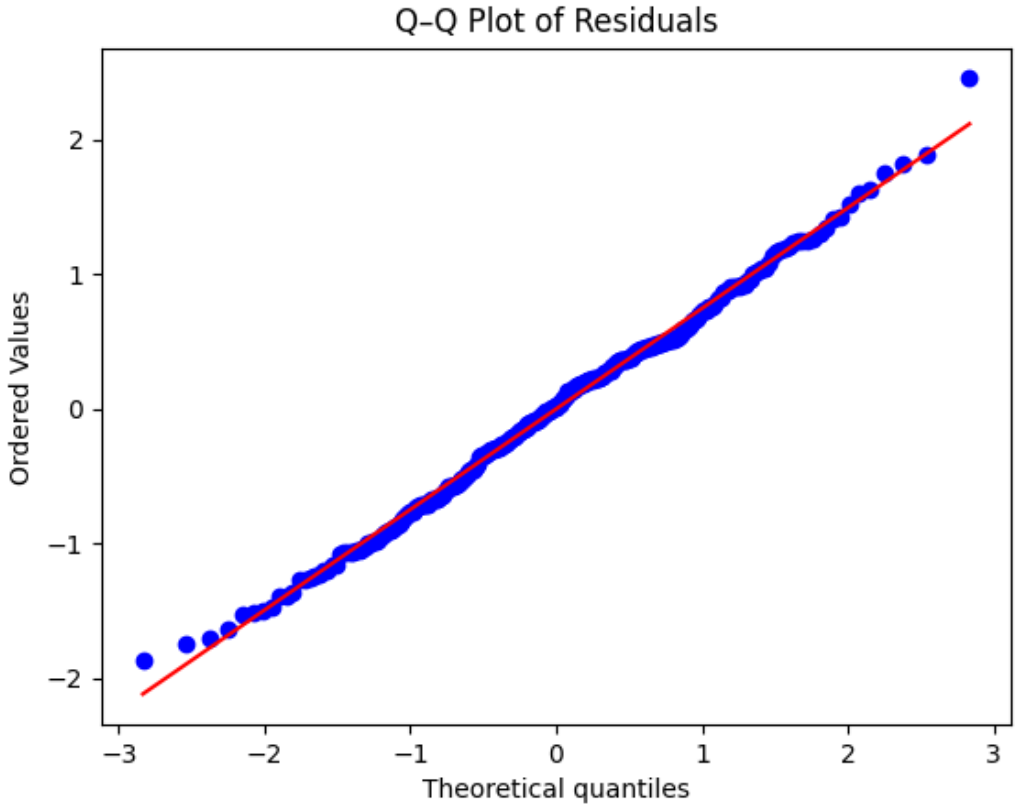


Figure 2: Q–Q Plot of Residuals

```
plt.show()
```

The Q-Q plot displays the standardized residuals against the theoretical quantiles of a standard normal distribution.
The points are following the 45-degree reference line extremely closely over the entire range(from -2 to +2) and there are very minor deviations at the extreme tails, but these are not of much significance.
Therefore, the residuals strongly follow a normal distribution, satisfying the normality assumption.

1.10 Part 10

1.11 Part 11

Violation	Indicated By	Inference on Model
Non-linearity	A curved pattern in the Residuals vs Fitted plot.	The given data is a curved pattern that can't be fit into a linear plot, requires a non-linear relationship.
Heteroscedasticity	A cone shape in the Residuals vs Fitted plot.	Variance is not the same everywhere, the error spread increases/decreases as the prediction ($\hat{y}$) gets larger.
Non-normality	Points deviating significantly from the straight line in the Q-Q plot.	The errors do not follow a Bell Curve, so we cannot trust the standard formulas that tell us if our results are statistically true or just random luck.

Table 1: Potential Violations of the Regression Assumptions

Listing 4: Identifying high-leverage and influential points.

```
pinvX = np.linalg.pinv(X)
H = X @ pinvX

# Leverage values
leverage = np.diag(H)

n, p_1 = X.shape
leverage_threshold = 2 * p_1 / n

high_leverage = np.where(leverage > leverage_threshold)[0]

MSE = np.mean(res**2)

cooks_d = (res**2/(p_1 * MSE)) * (leverage/(1 - leverage)**2)

cooks_threshold = 4/n
influential_points = np.where(cooks_d > cooks_threshold)[0]
```

The leverage h_{ii} measures how far an observation's predictor values (x_i) are from the center of the predictor space. A point is considered **high-leverage** if its x values are

far away from the mean of the x values. A common cutoff for h_{ii} is $2\times(p+1)/n$(where $p+1$ is the number of coefficients and n is the number of samples).

High-leverage points can have a large impact on the fitted regression line.

Points with **high Cook's distance** are called as **influential points**. A common rule of thumb for identifying them is $D_i > 1$ or $D_i > 4/n$. Where D_i is calculated as follows,

$$D_i = \frac{e_i^2}{s^2(p+1)}\left[\frac{h_{ii}}{(1-h_{ii})^2}\right]$$

1.12 Part 12

Consider the given expected value expression:

$$\mathbb{E}\big[(y-\hat{f}(x))^2\big]$$

Since $y = f(x) + \epsilon$,

$$\mathbb{E}\big[(f(x)+\epsilon-\hat{f}(x))^2\big]$$

$$\mathbb{E}\big[(f(x)-\hat{f}(x))^2+2\epsilon(f(x)-\hat{f}(x))+\epsilon^2\big]$$

Using linearity of expectation,

$$\mathbb{E}\big[(f(x)-\hat{f}(x))^2\big]+2\mathbb{E}\big[\epsilon(f(x)-\hat{f}(x))\big]+\mathbb{E}[\epsilon^2]$$

Since $\mathbb{E}[\epsilon]=0$ and ϵ is independent of $\hat{f}(x)$,

$$\mathbb{E}[\epsilon^2]=\sigma^2, \mathbb{E}\big[\epsilon(f(x)-\hat{f}(x))\big]=0$$

$$\therefore \mathbb{E}\big[(y-\hat{f}(x))^2\big]=\mathbb{E}\big[(f(x)-\hat{f}(x))^2\big]+\sigma^2$$

Now decompose the first term:

$$\mathbb{E}\big[(f(x)-\hat{f}(x))^2\big]=\big(\mathbb{E}[\hat{f}(x)]-f(x)\big)^2+\mathrm{Var}(\hat{f}(x))$$

Therefore,

$$\boxed{\mathbb{E}\big[(y-\hat{f}(x))^2\big]=\mathrm{Bias}^2+\mathrm{Var}+\sigma^2}$$

1.13 Part 13

Part(a)

We consider the model

$$y_i = \alpha_0 + \alpha_1 x_{1i} + u_i$$

which is fit to data generated by the true model

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \varepsilon_i$$

The OLS estimator minimizes

$$\sum_{i=1}^{n} (y_i - \alpha_0 - \alpha_1 x_{1i})^2$$

Taking partial derivatives and setting them to zero gives the normal equations. Differentiating with respect to α_0:

$$\sum_{i=1}^{n} (y_i - \alpha_0 - \alpha_1 x_{1i}) = 0,$$

$$\alpha_0 = \bar{y} - \alpha_1 \bar{x}_1 \tag{1}$$

Differentiating with respect to α_1:

$$\sum_{i=1}^{n} x_{1i}(y_i - \alpha_0 - \alpha_1 x_{1i}) = 0$$

Substituting $\alpha_0 = \bar{y} - \alpha_1 \bar{x}_1$,

$$\sum_{i=1}^{n} x_{1i}(y_i - \bar{y} + \alpha_1 \bar{x}_1 - \alpha_1 x_{1i}) = 0$$

$$\sum_{i=1}^{n} (x_{1i} - \bar{x}_1)(y_i - \bar{y}) = \alpha_1 \sum_{i=1}^{n} (x_{1i} - \bar{x}_1)^2 \tag{2}$$

From equations (1) and (2), [A well known formula]

$$\hat{\alpha}_1 = \frac{\sum_{i=1}^{n} (x_{1i} - \bar{x}_1)(y_i - \bar{y})}{\sum_{i=1}^{n} (x_{1i} - \bar{x}_1)^2}$$

Substituting the true model for y_i,

$$y_i - \bar{y} = \beta_1(x_{1i} - \bar{x}_1) + \beta_2(x_{2i} - \bar{x}_2) + (\varepsilon_i - \bar{\varepsilon})$$

Plugging this into the estimator,

$$\hat{\alpha}_1 = \beta_1 + \beta_2 \frac{\sum_{i=1}^n (x_{1i} - \bar{x}_1)(x_{2i} - \bar{x}_2)}{\sum_{i=1}^n (x_{1i} - \bar{x}_1)^2} + \frac{\sum_{i=1}^n (x_{1i} - \bar{x}_1)(\varepsilon_i - \bar{\varepsilon})}{\sum_{i=1}^n (x_{1i} - \bar{x}_1)^2}$$

Taking expectations and using $\mathbb{E}[\varepsilon_i \mid x_{1i}] = 0$, the last term vanishes. Therefore,

$$\boxed{\mathbb{E}[\hat{\alpha}_1] = \beta_1 + \beta_2 \frac{\mathrm{Cov}(x_1, x_2)}{\mathrm{Var}(x_1)}}$$

Part(b)

$$\mathrm{Bias}(\hat{\alpha}_1) = \mathbb{E}[\hat{\alpha}_1] - \beta_1$$

$$\mathrm{Bias}(\hat{\alpha}_1) = \beta_2 \frac{\mathrm{Cov}(x_1, x_2)}{\mathrm{Var}(x_1)}$$

Omitting x_2 has biased $\hat{\alpha}_1$.

Part(c)

- When covariance between x_1, x_2 is 0
- When variance of x_1 is much larger than the covariance between x_1 and x_2.

1.14 Part 14

The generation:

```
n = 500

# generating predictors
x1 = np.random.randn(n)
z = np.random.randn(n)
x2 = x1 + 0.9 * z

# Design matrix
X = np.column_stack((np.ones(n), x1, x2))
```

```
# Regression coefficients matrix
beta_true = np.array([3.0, -4.0, 7.0])
# noise
eps = np.random.randn(n)

# response
y = X @ beta_true + eps
```

Part(a)

```
# Condition number
XtX = X.T @ X
cond_number = np.linalg.cond(XtX)

print("Condition number of X^T X:", cond_number)
```

Part(b)

Since $\hat{\beta} = (X^T X)^{-1} X^T y$, On substituting y,

$$\hat{\beta} = (X^T X)^{-1} X^T (X\beta + \varepsilon) = \beta + (X^T X)^{-1} X^T \varepsilon$$

Then,

$$\mathrm{Var}(\hat{\beta}) = \mathrm{Var}\big((X^T X)^{-1} X^T \varepsilon\big)$$

Using $\mathrm{Var}(A\varepsilon) = A\,\mathrm{Var}(\varepsilon)\,A^T$,

$$\mathrm{Var}(\hat{\beta}) = (X^T X)^{-1} X^T \mathrm{Var}(\varepsilon) X (X^T X)^{-1}$$

Substituting $\mathrm{Var}(\varepsilon) = \sigma^2 I$,

$$\mathrm{Var}(\hat{\beta}) = \sigma^2 (X^T X)^{-1} X^T X (X^T X)^{-1} = \sigma^2 (X^T X)^{-1}$$

Therefore,

$$\boxed{\mathrm{Var}(\hat{\beta}) = \sigma^2 (X^T X)^{-1}}$$

```
alphas = [0.0, 0.4, 0.7, 0.9]

for a in alphas:
    x2_corr = x1 + a * np.random.randn(n)
    X_corr = np.column_stack((np.ones(n), x1, x2_corr))
    XtX_corr = X_corr.T @ X_corr
    var_beta = np.linalg.inv(XtX_corr)
    # Printed values to check in the real code file
```

As the correlation between x_1 and x_2 increases, the values of $\text{Var}(\hat{\beta}_1)$ and $\text{Var}(\hat{\beta}_2)$ increases.

Part(c)

Multicollinearity does not introduce bias because the OLS estimator remains unbiased:

$$\mathbb{E}[\hat{\beta}] = \beta,$$

However, multicollinearity inflates the variance of the estimator, leading to high sensitivity of coefficients to small changes in the data. Thus, multicollinearity causes instability but not bias.

2 Question 2: Salary Prediction & Bias Detection

2.1 Part 1

Since boxplots are the best for visualizing outliers, used them for salary across gender, education, and industry.

```python
# Part 1
df = pd.read_csv("salary_dataset.csv")

# Univariate Statistics
# Basic info
print(df.info())
print("-----------------------------------")
print(df.describe())
print("-----------------------------------")

# Salary distribution
plt.hist(df["salary"], bins=50)
plt.xlabel("Salary")
plt.ylabel("Frequency")
plt.title("Salary Distribution")
plt.show()

# Others
# Salary by gender
sns.boxplot(x="gender", y="salary", data=df)
plt.title("Salary by Gender")
plt.show()

# Salary by education
sns.boxplot(x="education_level", y="salary", data=df)
plt.title("Salary by Education Level")
plt.show()

# Salary by industry
sns.boxplot(x="industry", y="salary", data=df)
plt.title("Salary by Industry")
plt.show()

# Correlation heatmap (numerical features)
num_cols = df.select_dtypes(include=np.number)
sns.heatmap(num_cols.corr(), annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```

Figure 3: Statistics and Plotting

2.2 Part 2

```python
# Part 2
print("PART @2")

# Fill numeric misses with median
num_features = df.select_dtypes(include=np.number).columns
df[num_features] = df[num_features].fillna(df[num_features].median())

# Fill categorical misses with mode
cat_features = df.select_dtypes(exclude=np.number).columns
df[cat_features] = df[cat_features].fillna(df[cat_features].mode().iloc[0])

# Outlier handling using IQR for salary
Q1 = df["salary"].quantile(0.25)
Q3 = df["salary"].quantile(0.75)
IQR = Q3 - Q1

df = df[(df["salary"] >= Q1 - 1.5*IQR) & (df["salary"] <= Q3 + 1.5*IQR)]
```

Figure 4: Handling misses and outliers

Reasons for these particular decisions:

- For numeric features, mean would be a bad choice since it widely varies with the values of outliers(unstable), whereas median lies near the center of the distribution, so it would be a better choice to fill the misses.
- For categorial features, the most frequent category(mode) is the best choice because it preserves the majority distribution and avoids creating new categories.
- For outlier handling, I used the **Inter-quartile range(IQR)** which captures the middle 50% values, i.e, around the center of distribution, and values lying beyond $1.5 \times$ IQR from the quartiles are commonly treated as outliers.

2.3 Part 3

For categorial features, OneHotEncoder was used because the categorical features are nominal and lack any natural ordering. One-hot encoding converts categories into independent binary features, preventing the linear regression model from imposing

```python
# Part 3
print("PART @3")
X = df.drop("salary", axis=1)
y = df["salary"]

categorical_cols = X.select_dtypes(exclude=np.number).columns
numerical_cols = X.select_dtypes(include=np.number).columns

preprocessor = ColumnTransformer(
    transformers=[
        ("cat", OneHotEncoder(drop="first"), categorical_cols),
        ("num", "passthrough", numerical_cols)
    ]
)
print("PART @3 - DONE")
```

Figure 5: Encoding Categorial Features

artificial ordinal relationships or magnitudes which gives incorrect values in our model without knowing.
The *drop = "first"* is used because we do not want any multicollinearity/dependencies in those categories.
For numerical features, we just pass them as they are.

2.4 Part 4

```python
# Part 4
print("PART @4")
# Stratify by gender
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=X["gender"], random_state=42
)
print("PART @4 - DONE")
```

Figure 6: Stratification

Gender was chosen as the stratification variable because it is treated as a protected attribute in this analysis. Stratifying on gender ensures that all gender groups are proportionally represented in both the training and test sets, which is necessary for

reliable evaluation of model performance and fairness metrics across demographic groups.

2.5 Part 7

Although the dataset contains 11 original predictors, categorical variables were expanded into multiple binary features via one-hot encoding, resulting in a larger number of regression coefficients corresponding to individual categories.

```
PART @7
|       |       |      Feature   Coefficient  P-value
const                Intercept  51080.067354      0.0
x32    cat__city_San Francisco  41767.621511      0.0
x30         cat__city_New York  37412.880392      0.0
x28       cat__city_Los Angeles  24173.111013      0.0
x33          cat__city_Seattle  23888.041688      0.0
PART @7 - DONE
```

Figure 7: Values and Interpretations

Interpretations:

- The intercept representing the baseline predicted salary for an employee belonging to the reference categories of all categorical variables(with numerical features set to zero) is the highest. As this scenario is not practically meaningful, the intercept can't be interpreted further.
- The remaining coefficients correspond to city-level indicators. The positive coefficients indicate that employees working in San Francisco earn significantly higher salaries compared to the reference city, holding all other variables constant. New York also shows a strong positive effect, followed by Los Angeles and Seattle, suggesting that employees in these cities earn higher salaries relative to the baseline location.

Note: The reference city is the category omitted during one-hot encoding and serves as the baseline against which the salary effects of other cities are measured.

2.6 Part 9

The residuals versus fitted values plot shows residuals randomly scattered around zero with no strong systematic pattern, suggesting that the linearity assumption

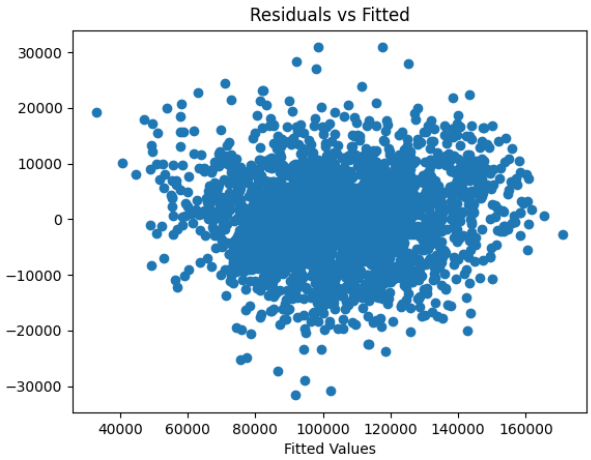


Figure 8: Residuals vs Fitted Values

is reasonably satisfied. The spread of residuals appears mostly constant across fitted values, a slightly larger spread around the median of fitted values suggests the presence of mild heteroscedasticity.

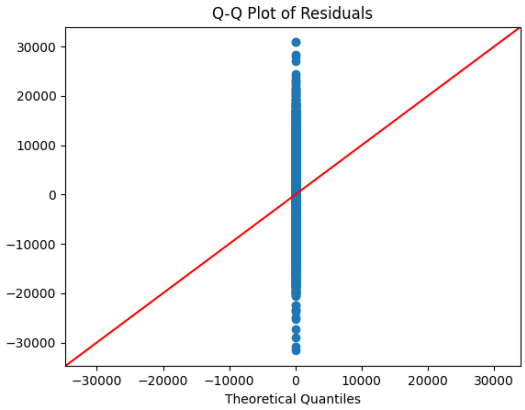


Figure 9: Q-Q Plot

The Q-Q plot of residuals shows substantial deviations from the diagonal line, indicating that the residuals do not follow a normal distribution. The presence of heavy tails suggests that large prediction errors occur more frequently than those under normality of errors.

2.7 Part 10

Metric Definitions

To evaluate the fairness of the model with respect to gender, the following six metrics were computed[cite: 53, 54, 55, 56, 57, 58]:

- **(a) Mean Salary Prediction Difference:** The difference in the average predicted salary between two groups.
- **(b) Mean Absolute Error (MAE) per group:** Measures the average magnitude of prediction errors for a specific group. Comparing MAE helps determine if the model is more accurate for one group than another.
- **(c) Demographic Parity Difference (DPD):** The difference in the selection rate(probability of being predicted as a "high earner") between groups.
- **(d) Equal Opportunity Difference (EOD):** The difference in True Positive Rates (TPR) between groups. It measures whether qualified individuals(high actual salary) in different groups have an equal chance of being correctly predicted as high earners.
- **(e) Predictive Equality:** The difference in **False Positive Rates** (FPR) between groups. It measures whether unqualified individuals(low actual salary) in different groups have an equal probability of being incorrectly predicted as high earners.
- **(f) Disparate Impact Ratio (DIR):** The ratio of the selection rates between the disadvantaged group and the privileged group. A value close to 1 indicates fairness, while values significantly below 1 indicate disparate impact.

Results and Bias Interpretation

- **Disparity in Predictions:** The Mean Salary Prediction Difference is positive (favoring Males), indicating that the model, on average, predicts higher salaries for Male employees compared to Female and Other groups.
- **Demographic Parity:** The positive DPD and a DIR deviating from 1.0 (Male/Female) suggest that the model selects Males for the "high earner" category at a significantly higher rate than Females and Others.

- **Conclusion on Bias:** Based on these metrics, the model exhibits **systematic bias against Female and Other genders**. The disparity suggests the model has learned historical biases present in the training data.

2.8 Part 12

```
gender
Female    -221.039938
Male        73.846912
Other     2514.193446
Name: residual, dtype: float64
['Overestimated', 'Underestimated', 'Underestimated']
```

Figure 10: Estimation

Since positive mean residuals indicate underestimation of salaries, while negative mean residuals indicate overestimation,

- Female group is **Overestimated**
- Male and Other groups are **Underestimated**

2.9 Part 13

Interpretations:

As shown in the SHAP Summary Plot

- **Dominant Feature:** $'num\_years\_experience'$ is the most influential predictor. The color gradient clearly shows that higher experience leads to significantly higher predicted salaries.
- **Gender Impact:** $'cat\_gender\_Male'$ appears as a top feature with a clear split. The red points (Male=1) have positive SHAP values, confirming the model adds a premium to the prediction for being Male.
- **Education:** Higher education levels (e.g., PhD, Masters) generally contribute positively, while 'HighSchool' (as seen in the dependence plot) has a strong negative impact.

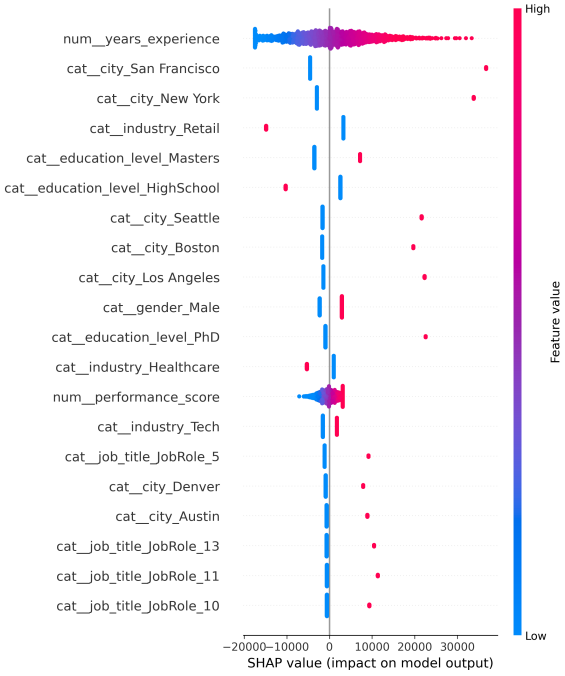
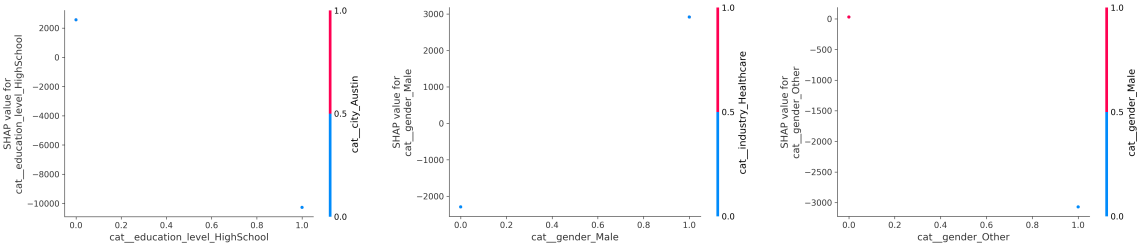


Figure 11: Summary Plot



(a) Education_HighSchool (b) Gender_Male (c) Gender_Other

Figure 12: Dependence plots

Dependence Analysis

Dependence plots for the top 3 features confirm:

1. **Gender_Male:** Shows a binary cluster where '1' (Male) consistently adds positive value to the output compared to '0' by other 2 groups, indicating a higher salary of Males.

2. **Education (High School):** Shows a large negative contribution, indicating a penalty for having a low education.

3 Question 3: Deep Neural Network Classifier for Hand-written Digits

3.1 Part 1

ReLU is preferred over Sigmoid and Tanh in deep networks because,

- Sigmoid and Tanh suffer from the vanishing gradient problem, since their derivatives approach zero for large positive or negative inputs. This causes gradients to diminish as they are backpropagated through many layers, slowing or preventing learning in deep networks. In contrast, the ReLU function

$$\text{ReLU}(x) = \max(0, x)$$

  has a constant gradient for positive inputs, which helps preserve gradient magnitude and enables effective training of deep models.

- ReLU is computationally simpler and more efficient than Sigmoid and Tanh, as it involves only a thresholding operation rather than exponential functions. This results in faster training and encourages sparse activations, which can improve optimization and generalization.

3.2 Part 2

PyTorch's autograd engine provides automatic differentiation by dynamically building a computation graph during the forward pass. Each operation on tensors that require gradients is recorded as a node in this graph, with edges representing the flow of data.
During backpropagation, triggered by calling `loss.backward()`, autograd traverses the computation graph in reverse order and applies the chain rule to compute gradients of the loss with respect to each parameter. These gradients are accumulated in the `.grad` attribute of each learnable parameter, enabling efficient optimization without requiring manual derivative calculations.Thus helping the developer a lot!.