PRESENTED BY:
GROUP 5 4CSC
GUEVARRA
JAVIER
LUCES
ROMANES
TAPAO

DATA ANALYSIS AND VISUALIZATION
14 DECEMBER 2024

SALARY DECODED

A Data-Driven Analysis of IT Compensation

CONTENT

STATEMENT OF THE PROBLEM

IT professionals **often face challenges in determining fair compensation** due to the rapid evolution of technology and the diverse range of roles in the industry.

The absence of clear, data-driven benchmarks creates:

- Uncertainty in salary negotiations.
- Recruitment inefficiencies leading to mismatches.
- Difficulty for businesses to structure competitive salary packages.

OBJECTIVE

To **develop a machine learning-based salary prediction model** that provides IT professionals with accurate, data-driven salary estimates.

The model will:

- Account for factors like experience, specialization, skills, location, and education.
- Empower professionals to negotiate with confidence.
- Help employers create competitive salary structures that attract and retain talent.

DATASET OVERVIEW

TITLE: IT Jobs in the Asia-Pacific Region (May-June 2024)
SOURCE: Kaggle
SCOPE: IT job vacancies across the Asia-Pacific region (Australia, Indonesia, Malaysia, Philippines, Singapore, Hong Kong)

FILES	
itjob_main.csv	Primary information about each job listing
itjob_header.csv	Primary information about the employee
itjob_tools.csv	Required tools or technologies for jobs
itjob_prog_lang.csv	Programming languages needed for jobs
itjob_certification.csv	Mandatory/non-mandatory certifications
itjob_main_spec.csv	Additional specializations required

DATASET OVERVIEW

The dataset contains several job listings and features that help categorize and detail each job opening in the region

KEY FEATURES	
jobid:	Unique job identifier
country:	Location of the job (country-specific)
level:	Job level (e.g., Junior, Senior)
specialization:	Job area of focus (e.g., Software Engineering, Data Science)
salary_from / salary_to:	Salary range offered
currency:	Currency used in salary (e.g., USD, SGD)
employment type:	Full-time, Part-time, etc.
work mode:	Work mode options like Hybrid, Remote, Office
visa_sponsorship:	Availability of visa sponsorship
work_experience_years:	Required years of work experience
education_level:	Required education level (e.g., Bachelor's, Master's)

DATA ANALYSIS PROCESS

DATA EXPLORATION

Dataset Review:
Examined relevant files to understand data structure and content.

Data Type Conversion: Converted columns from object to string for consistency and accuracy.

Relevance Filtering: Excluded files with irrelevant or overly unique values.

Standardization: Standardized certain values to ensure equal scale and treatment across the dataset.

```
# Load itjob_prog_lang.csv
prog_lang = pd.read_csv("/content/drive/MyDrive/Project/archiv

# Convert object data type column to string
prog_lang['prog_lang_text'] = prog_lang['prog_lang_text'].asty

# Check info
prog_lang.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34245 entries, 0 to 34244
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   prog_lang_id    34245 non-null  int64
 1   jobid           34245 non-null  int64
 2   prog_lang_text  34239 non-null  string
```

```
# Load itjob_cert.csv
cert = pd.read_csv("/content/drive/MyDrive/Project/arch

# Convert object data type column to string
cert['certification_text'] = cert['certification_text']

# Check info
cert.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14087 entries, 0 to 14086
Data columns (total 4 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   cert_id            14087 non-null  int64
 1   jobid              14087 non-null  int64
 2   certification_text 14084 non-null  string
 3   is_mandatory       14087 non-null  int64
dtypes: int64(3), string(1)
```

```
# Load itjob_header.csv
header = pd.read_csv("/content/drive/MyDrive/Project/archive/itj

# Convert object data type column to string
header = header.astype({col: 'string' for col in header.select_d

# Check info
header.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32839 entries, 0 to 32838
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   jobid                 32839 non-null  int64
 1   level                 27852 non-null  string
 2   tech_specialisation   29648 non-null  string
 3   country               32808 non-null  string
 4   state                 8000 non-null   string
 5   salary_from           9407 non-null   float64
 6   salary_to             8664 non-null   float64
 7   currency              9179 non-null   string
 8   type                  32828 non-null  string
 9   mode                  7947 non-null   string
 10  visa_sponsorship      838 non-null    string
 11  work_experience_years 19540 non-null  float64
 12  education_level       18118 non-null  string
```

```
# Standardize currency
# Php and PHP (both philippine peso)
# Baht and THB (both Thai Baht)
# IDR and Rp (both Indonesian Rupiah)
# RM and MYR (both Malaysian Ringgit)

# Define a mapping dictionary for currency standardization
currency_mapping = {
    'PHP': 'PHP', 'Php': 'PHP',  # Philippine Peso
    'THB': 'THB', 'Baht': 'THB', ' THB': 'THB',   # Thai Baht # Handling extra spaces as well
    'IDR': 'IDR', 'Rp': 'IDR',    # Indonesian Rupiah
    'MYR': 'MYR', 'RM': 'MYR'     # Malaysian Ringgit
}

# Apply the mapping to standardize specific currency values
header['currency'] = header['currency'].replace(currency_mapping)

# Standardize and capitalize all currency values
header['currency'] = header['currency'].str.strip().str.upper()

# Check the unique values after standardization
print(header['currency'].unique())
```

```
<StringArray>
[ <NA>, 'PHP', 'SGD', 'AUD', 'NZD', 'MYR', 'IDR', 'HKD', 'THB', 'USD', 'GBP',
 'JPY', 'FJD', 'MXN']
```

```
# Load itjob_main.csv
main = pd.read_csv("/content/drive/MyDrive/Project/archive/itjob_main.csv")

main['source_classification'] = main['source_classification'].astype('string')

# Check info
main.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3717 entries, 0 to 3716
Data columns (total 2 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   jobid                3717 non-null   int64
 1   source_classification  3717 non-null   string
dtypes: int64(1), string(1)
```

```
# Standardize type
# Standardize and capitalize all type values
header['type'] = header['type'].str.upper()

# Check the unique values after standardization
type_unique_standardized = header['type'].unique()
print(type_unique_standardized)
```

```
<StringArray>
[      'FULL TIME',    'CONTRACT/TEMP',       'INTERNSHIP',          'PART
 'CASUAL/VACATION',              <NA>,        'VOLUNTEER']
Length: 7, dtype: string
```

DATA ANALYSIS PROCESS

DATA CLEANING & PREPROCESSING

Dataset Merging:

Combined **header**, **main**, and **prog_lang** datasets based on jobid to create a unified dataset.

Handling Missing Values:

Dropped rows with missing **salary_from** and **salary_to**.

Removed null entries in **prog_lang_text**, **tech_specialisation**, **currency**, **country**, and **work_experience_years**.

Salary Standardization:

Converted salaries to USD, added columns **salary_from_usd** and **salary_to_usd**.

Removed original salary columns.

Additional Merging:

Merged with "certification_text" from the certification dataset using **jobid**.

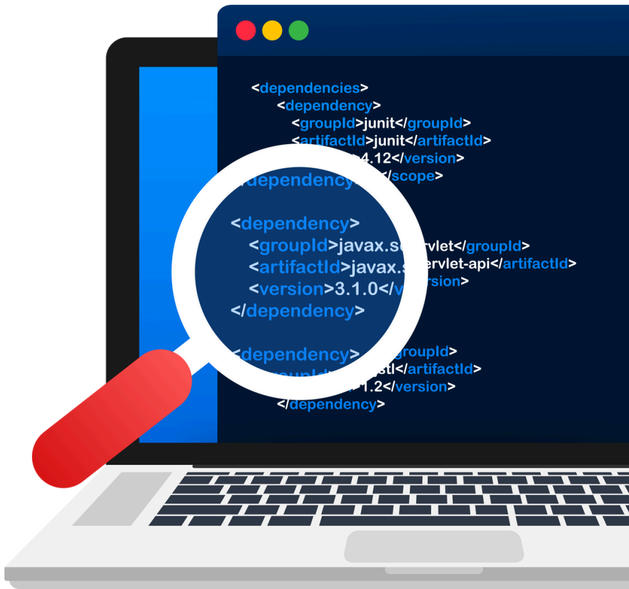
Imputation for Missing Features:

Predicted and filled missing **level** and **education_level** using Random Forest Classifier to retain key features.

Feature Removal:

Dropped irrelevant or null-heavy columns: **mode**, **visa_sponsorship**, **source_classification**, **certification_text**.

Results:
Streamlined dataset with complete, normalized, and relevant features for analysis and modeling.



DATA ANALYSIS PROCESS

```python
# List of exchange rates
exchange_rates = {
    'PHP': 0.018,   # Philippine Peso to USD
    'SGD': 0.73,    # Singapore Dollar to USD
    'AUD': 0.64,    # Australian Dollar to USD
    'MYR': 0.21,    # Malaysian Ringgit to USD
    'IDR': 0.000065, # Indonesian Rupiah to USD
    'HKD': 0.13,    # Hong Kong Dollar to USD
    'THB': 0.027,   # Thai Baht to USD
    'NZD': 0.59,    # New Zealand Dollar to USD
    'USD': 1.0,     # US Dollar to USD (no conversion)
    'GBP': 1.22,    # British Pound to USD
    'JPY': 0.0067,  # Japanese Yen to USD
    'FJD': 0.44,    # Fijian Dollar to USD
    'MXN': 0.055    # Mexican Peso to USD
}

def convert_to_usd(salary, currency):
    if pd.notna(salary) and currency in exchange_rates:
        return salary * exchange_rates[currency]
    return salary  # In case of missing or unknown currency, return original salary

# Apply the conversion function to salary_from and salary_to columns
have_salary['salary_from_usd'] = have_salary.apply(lambda row: convert_to_usd(row['salary_from'], row['currency']), axis=1)
have_salary['salary_to_usd'] = have_salary.apply(lambda row: convert_to_usd(row['salary_to'], row['currency']), axis=1)
```

```python
# Count the occurrences of each category in the 'level' column
level_counts = have_level['level'].value_counts()

# Display the counts for each category
print(level_counts)
```

```
0
<StringArray>
['Junior', 'Middle', 'Senior', 'Lead']
Length: 4, dtype: string
<class 'pandas.core.frame.DataFrame'>
Index: 6174 entries, 7 to 16429
Data columns (total 15 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   jobid                  6174 non-null   int64
 1   level                  6174 non-null   string
 2   tech_specialisation    6174 non-null   string
 3   country                6174 non-null   string
 4   currency               6174 non-null   string
 5   type                   6174 non-null   string
 6   mode                   1900 non-null   string
 7   visa_sponsorship       184 non-null    string
 8   work_experience_years  6174 non-null   float64
 9   education_level        4058 non-null   string
 10  source_classification  667 non-null    string
 11  prog_lang_text         6174 non-null   string
 12  salary_from_usd        6174 non-null   float64
 13  salary_to_usd          6174 non-null   float64
 14  certification_text     1351 non-null   string
dtypes: float64(3), int64(1), string(11)
memory usage: 771.8 KB
None
level
Middle    3346
Senior    2229
Junior     378
Lead       221
Name: count, dtype: Int64
```

```python
# Merge header and main on 'jobid'
merged_df = pd.merge(header, main, on='jobid', how='left')

# Merge the result with prog_lang on 'jobid'
merged_df = pd.merge(merged_df, prog_lang, on='jobid', how='left')

merged_df.info()
merged_df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54097 entries, 0 to 54096
Data columns (total 16 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   jobid                  54097 non-null  int64
 1   level                  46720 non-null  string
 2   tech_specialisation    50229 non-null  string
 3   country                54059 non-null  string
 4   state                  11423 non-null  string
 5   salary_from            15015 non-null  float64
 6   salary_to              13976 non-null  float64
 7   currency               14599 non-null  string
 8   type                   54084 non-null  string
 9   mode                   13039 non-null  string
 10  visa_sponsorship       1350 non-null   string
 11  work_experience_years  32813 non-null  float64
 12  education_level        31876 non-null  string
 13  source_classification  5973 non-null   string
 14  prog_lang_id           34225 non-null  float64
 15  prog_lang_text         34219 non-null  string
dtypes: float64(4), int64(1), string(11)
memory usage: 6.6 MB
```

```python
# Merge the current dataset with two columns from cert dataset
have_salary = pd.merge(have_salary, cert[['jobid', 'certification_text']], on='jobid', how='left')

# Drop prog_lang_id column and drop entries with null in prog_lang_text
have_prog_lang = have_salary.drop('prog_lang_id', axis=1).dropna(subset=['prog_lang_text'])

# Drop entries with null in tech specialization
have_tech_specialization = have_prog_lang.dropna(subset=['tech_specialisation'])

# Retain 'tech_specialization' and drop 'state' column
have_tech_specialization_no_state = have_tech_specialization.drop('state', axis=1)

# Drop null entries in the dataset from 'currency' column
have_currency = have_tech_specialization_no_state.dropna(subset=['currency'])

# Drop entries with no country
have_country = have_currency.dropna(subset=['country'])

# Drop entries with null work experience years
have_work_exp = have_country.dropna(subset=['work_experience_years'])
have_work_exp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 6174 entries, 7 to 16429
Data columns (total 15 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   jobid                  6174 non-null   int64
 1   level                  5695 non-null   string
 2   tech_specialisation    6174 non-null   string
 3   country                6174 non-null   string
 4   currency               6174 non-null   string
 5   type                   6174 non-null   string
 6   mode                   1900 non-null   string
 7   visa_sponsorship       184 non-null    string
 8   work_experience_years  6174 non-null   float64
 9   education_level        4058 non-null   string
 10  source_classification  667 non-null    string
 11  prog_lang_text         6174 non-null   string
 12  salary_from_usd        6174 non-null   float64
 13  salary_to_usd          6174 non-null   float64
 14  certification_text     1351 non-null   string
dtypes: float64(3), int64(1), string(11)
memory usage: 771.8 KB
```

```python
# Create a copy of the original DataFrame to preserve the original
have_level = have_work_exp.copy()

# Identify rows with non-null values for 'level'
train_data = have_level.dropna(subset=['level'])  # Rows with non-null 'level' for training

# Encode the 'level' column (target) for model compatibility
label_encoder = LabelEncoder()
train_data = train_data.copy()  # Explicitly create a copy to avoid warnings
train_data['level_encoded'] = label_encoder.fit_transform(train_data['level'])

# Prepare features (work_experience_years) and target (level_encoded) for the model
X_train = train_data[['work_experience_years']]  # Use 'work_experience_years' to predict 'level'
y_train = train_data['level_encoded']  # Encoded 'level' for training

# Train a Random Forest classifier
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Identify rows with missing values for 'level'
missing_data = have_level[have_level['level'].isnull()]
X_missing = missing_data[['work_experience_years']]  # Use the same feature for missing rows

# Predict missing 'level' values (encoded)
predicted_levels_encoded = rf_model.predict(X_missing)

# Convert the predicted encoded values back to original categories
predicted_levels = label_encoder.inverse_transform(predicted_levels_encoded)

# Impute the missing 'level' values in the original DataFrame (have_level)
have_level.loc[have_level['level'].isnull(), 'level'] = predicted_levels

# Check if there are still missing values in 'level'
print(have_level['level'].isnull().sum())  # Should be 0 after imputation

# Display unique values in 'level' after imputation
print(have_level['level'].unique())

print(have_level.info())
```

```python
# Getting only the entries that are not null in both salary_from and salary_to
have_salary = merged_df.dropna(subset=['salary_from', 'salary_to'])
display(have_salary)
have_salary.shape
```

DATA ANALYSIS PROCESS

```
# Standardize education_level column
have_educ = have_level.copy()

education_mapping = {
    'Diploma': 'Diploma/Certificate',
    'Certificate': 'Diploma/Certificate',
    'Associate Degree': 'Associate Degree',
    'Bachelor Degree': 'Bachelor Degree',
    'College Degree': 'Bachelor Degree',
    'Degree (Any)': 'Bachelor Degree',
    "Master's Degree": "Master's Degree",
    'Ph.D.': 'Ph.D.'
}

have_educ['education_level'] = have_educ['education_level'].map(education_mapping)

# Encode 'education_level' and 'level' for model compatibility (only for non-null rows in have_educ)
train_data = have_educ.dropna(subset=['education_level']).copy()  # Create a copy to avoid warnings

# Encode the 'level' column for model compatibility (before using it for imputation)
label_encoder_level = LabelEncoder()
train_data['level_encoded'] = label_encoder_level.fit_transform(train_data['level'])

# Encode 'education_level' for model compatibility (only for non-null rows in have_educ)
label_encoder = LabelEncoder()
train_data['education_level_encoded'] = label_encoder.fit_transform(train_data['education_level'])

# Prepare features (level and work_experience_years) and target (education_level_encoded)
X_train = train_data[['level_encoded', 'work_experience_years']]  # Features for training
y_train = train_data['education_level_encoded']  # Target for training

# Train a Random Forest classifier for imputation
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Identify rows with missing values for 'education_level' in have_educ
missing_data = have_educ[have_educ['education_level'].isnull()]  # Rows with missing 'education_level'

# Debugging step: Print the number of rows with missing 'education_level'
print(f"Number of rows with missing 'education_level': {missing_data.shape[0]}")
```

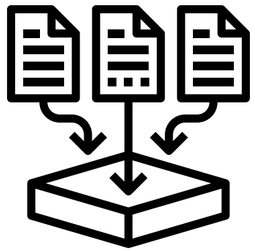
```
# If there are rows with missing values, proceed with imputation
if missing_data.shape[0] > 0:
    # Prepare features for the rows with missing 'education_level'
    X_missing = missing_data[['level', 'work_experience_years']].copy()

    # Debugging step: Check if X_missing has data
    print(f"Rows to be used for imputation: {X_missing.shape[0]}")

    # Encode 'level' in the rows with missing education_level
    X_missing['level_encoded'] = label_encoder_level.transform(X_missing['level'])  # Encode 'level' for missing rows

    # Predict missing 'education_level' values (encoded)
    predicted_education_level_encoded = rf_model.predict(X_missing[['level_encoded', 'work_experience_years']])

    # Convert the predicted encoded values back to original 'education_level' categories
    predicted_education_level = label_encoder.inverse_transform(predicted_education_level_encoded)

    # Impute the missing 'education_level' values in have_educ
    have_educ.loc[have_educ['education_level'].isnull(), 'education_level'] = predicted_education_level

    # Check if there are still missing values in 'education_level' in have_educ
    print(f"Missing values after imputation: {have_educ['education_level'].isnull().sum()}")
else:
    print("No missing values found for 'education_level'.")

# Convert 'educ_type' column to string
have_educ['education_level'] = have_educ['education_level'].astype('string')

# Verify the change
print(have_educ['education_level'].dtype)  # Should print 'object' or 'str'

# Display the updated DataFrame info for have_educ
have_educ.info()

# Count the occurrences of each category in the 'level' column
educ_counts = have_educ['education_level'].value_counts()

# Display the counts for each category
print(educ_counts)
```

```
# Remove remaining null values
# Count null values per column
null_counts = have_educ.isnull().sum()

# Display null counts
print("Null values per column:")
print(null_counts)

# Drop the 'mode' and 'visa_sponsorship' columns entirely
final_data = have_educ.drop(columns=['mode', 'visa_sponsorship','source_classification','certification_text'], errors='ignore')
```

```
Null values per column:
jobid                    0
level                    0
tech_specialisation      0
country                  0
currency                 0
type                     0
mode                  4274
visa_sponsorship      5990
work_experience_years    0
education_level          0
source_classification 5507
prog_lang_text           0
salary_from_usd          0
salary_to_usd            0
certification_text    4823
dtype: int64
```

```
Number of rows with missing 'education_level': 2116
Rows to be used for imputation: 2116
Missing values after imputation: 0
string
<class 'pandas.core.frame.DataFrame'>
Index: 6174 entries, 7 to 16429
Data columns (total 15 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   jobid                  6174 non-null   int64
 1   level                  6174 non-null   string
 2   tech_specialisation    6174 non-null   string
 3   country                6174 non-null   string
 4   currency               6174 non-null   string
 5   type                   6174 non-null   string
 6   mode                   1900 non-null   string
 7   visa_sponsorship       184 non-null    string
 8   work_experience_years  6174 non-null   float64
 9   education_level        6174 non-null   string
 10  source_classification  667 non-null    string
 11  prog_lang_text         6174 non-null   string
 12  salary_from_usd        6174 non-null   float64
 13  salary_to_usd          6174 non-null   float64
 14  certification_text     1351 non-null   string
dtypes: float64(3), int64(1), string(11)
memory usage: 771.8 KB
education_level
Bachelor Degree         5572
Diploma/Certificate      546
Master's Degree           42
Associate Degree           8
Ph.D.                      6
Name: count, dtype: Int64
```

DATA ANALYSIS PROCESS



DESCRIPTIVE ANALYSIS

Average Salary Calculation:

Derived average salary using the mean of "**salary_from_usd**" and "**salary_to_usd**" to understand its relationship with other variables.

Dataset Finalization:

Reviewed and validated the final dataset to ensure readiness for analysis.

Descriptive Analysis:

Summarized key data characteristics (e.g., distributions, central tendencies) to establish a foundation for deeper exploration.

DESCRIPTIVE STATISTICS KEY POINTS

Work Experience:

Average: 3.7 years, ranging from 0 to 15 years.

Salary Ranges:

Average starting salary: $11,791.
Average upper salary: $14,791.
High variability indicated by standard deviations.

Data Quality Check:

Minimum salary value of 0 suggests potential data entry errors or missing data requiring investigation.

Quartiles:

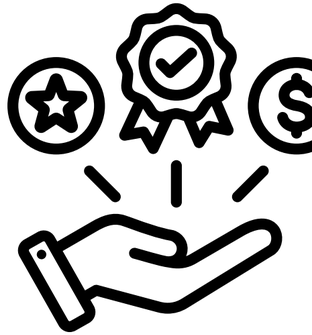
Insights into salary and experience distribution (25th, 50th, 75th percentiles).

Model Preparation:

Variability highlights the importance of feature scaling for accurate salary predictions.

```python
# Calculate the average salary in USD
final_data['average_salary_usd'] = (final_data['salary_from_usd'] + final_data['salary_to_usd']) / 2

final_data['average_salary_usd']


# Display the basic structure of the dataset
print("Basic Info:")
print(final_data.info())

# Descriptive statistics for numerical columns
print("\nDescriptive Statistics for Numerical Features:")
print(final_data.describe())

# Check for missing values
print("\nMissing Values:")
print(final_data.isnull().sum())
```

```
Basic Info:
<class 'pandas.core.frame.DataFrame'>
Index: 6174 entries, 7 to 16429
Data columns (total 11 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   jobid                  6174 non-null   int64
 1   level                  6174 non-null   string
 2   tech_specialisation    6174 non-null   string
 3   country                6174 non-null   string
 4   currency               6174 non-null   string
 5   type                   6174 non-null   string
 6   work_experience_years  6174 non-null   float64
 7   education_level        6174 non-null   string
 8   prog_lang_text         6174 non-null   string
 9   salary_from_usd        6174 non-null   float64
 10  salary_to_usd          6174 non-null   float64
dtypes: float64(3), int64(1), string(7)
memory usage: 578.8 KB
None

Descriptive Statistics for Numerical Features:
              jobid  work_experience_years  salary_from_usd  salary_to_usd
count   6174.000000            6174.000000      6174.000000    6174.000000
mean   15328.464367               3.683835     11791.373207   14791.517750
std     9481.312745               2.233598     25698.155446   31836.534459
min       16.000000               0.000000         0.000000       2.860000
25%     7048.000000               2.000000       936.000000    1350.000000
50%    14794.000000               3.000000      1890.000000    2626.500000
75%    23606.000000               5.000000      4550.000000    6205.000000
max    32855.000000              15.000000    195000.000000  256000.000000

Missing Values:
jobid                    0
level                    0
tech_specialisation      0
country                  0
currency                 0
type                     0
work_experience_years    0
education_level          0
prog_lang_text           0
salary_from_usd          0
salary_to_usd            0
dtype: int64
```



DATA ANALYSIS PROCESS

DISTRIBUTION OF FEATURES

Work Experience:
Right-skewed distribution with most data clustered at lower experience levels which may result in higher model accuracy for individuals with less experience.

Salary Range:
Potential skewness in salary features (**salary_from_usd**, **salary_to_usd**) indicating potential bias in predictions

Model Considerations:
Skewed distributions highlight the importance of feature engineering to ensure accuracy and fairness across all values.

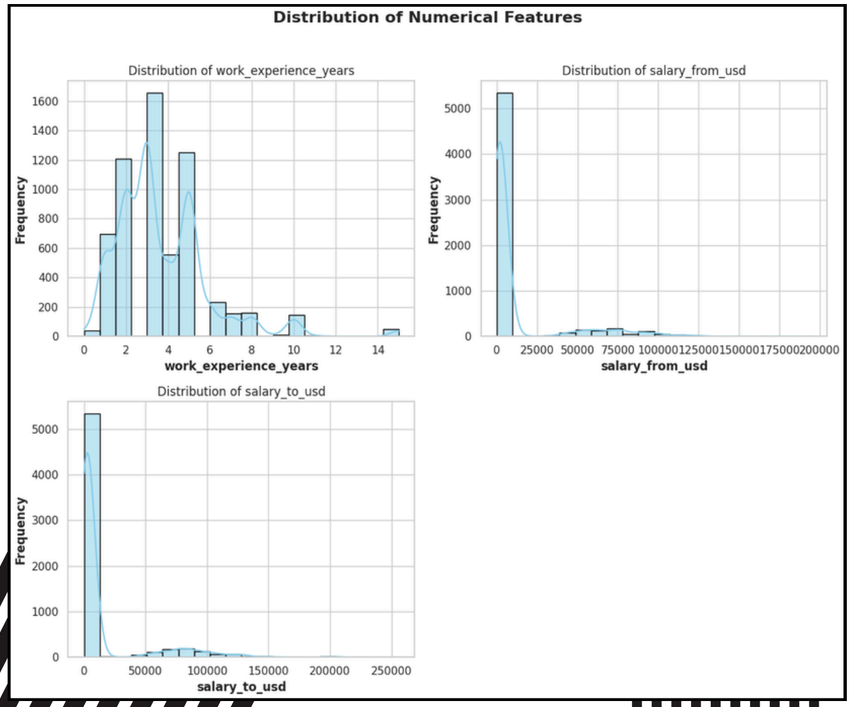
CATEGORICAL FEATURE INSIGHTS

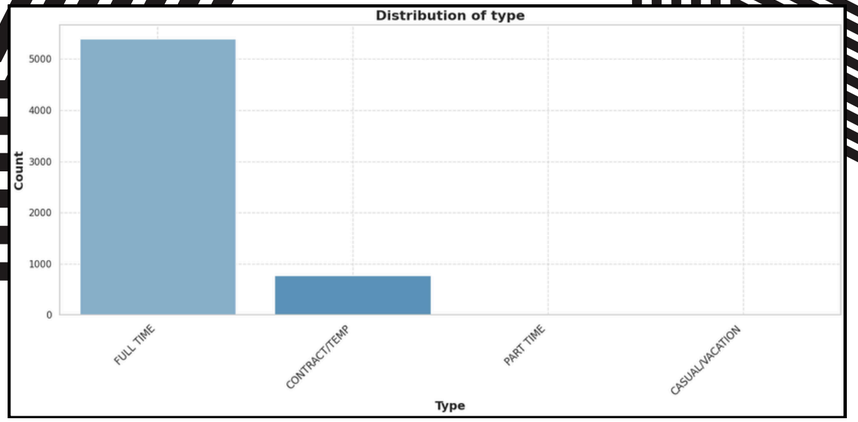
Data Imbalances:
Over-representation of middle-level jobs in "level" feature suggests potentially skewed predictions toward over-represented groups. This also highlights the importance of certain features as a predictor.

Encoding Considerations:
Selection of encoding methods for categorical feature impacts model accuracy and interpretability.

Model Complexity:
High number of unique categories adds to model complexity.

Feature Engineering:
Combining visualizations with domain knowledge ensures accurate and unbiased salary predictions.





DATA ANALYSIS PROCESS

DISTRIBUTION OF FEATURES

Work Experience:
Right-skewed distribution with most data clustered at lower experience levels which may result in higher model accuracy for individuals with less experience.

Salary Range:
Potential skewness in salary features (**salary_from_usd**, **salary_to_usd**) indicating potential bias in predictions

Model Considerations:
Skewed distributions highlight the importance of feature engineering to ensure accuracy and fairness across all values.

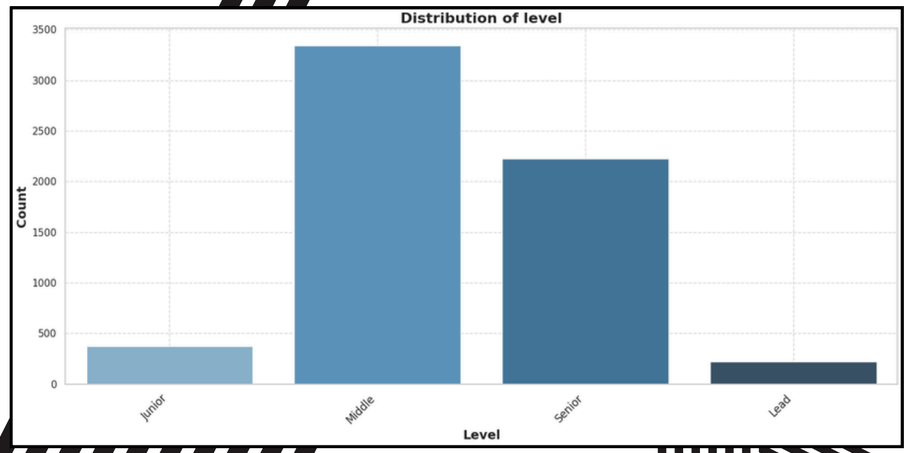
CATEGORICAL FEATURE INSIGHTS

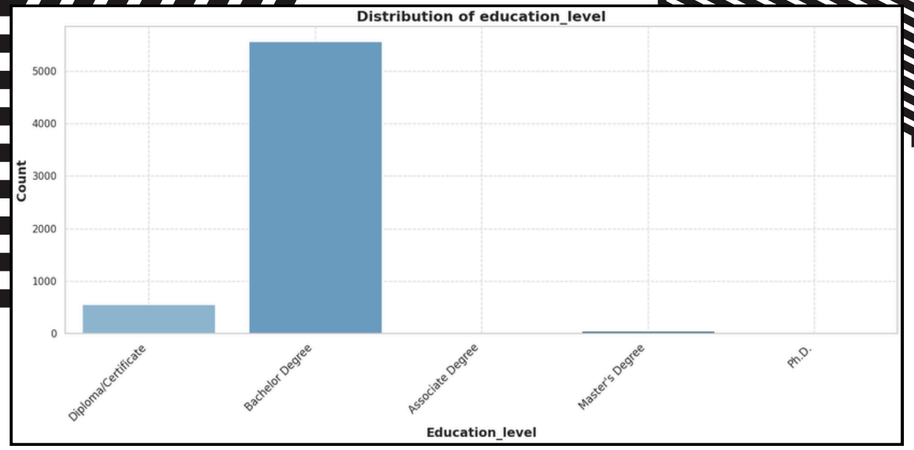
Data Imbalances:
Over-representation of entry-level jobs in "level" feature suggests potentially skewed predictions toward over-represented groups. This also highlights the importance of certain features as a predictor.

Encoding Considerations:
Selection of encoding methods for categorical feature impacts model accuracy and interpretability.

Model Complexity:
High number of unique categories adds to model complexity.

Feature Engineering:
Combining visualizations with domain knowledge ensures accurate and unbiased salary predictions.





DATA ANALYSIS PROCESS

DISTRIBUTION OF FEATURES

Work Experience:
Right-skewed distribution with most data clustered at lower experience levels which may result in higher model accuracy for individuals with less experience.

Salary Range:
Potential skewness in salary features (**salary_from_usd**, **salary_to_usd**) indicating potential bias in predictions

Model Considerations:
Skewed distributions highlight the importance of feature engineering to ensure accuracy and fairness across all values.

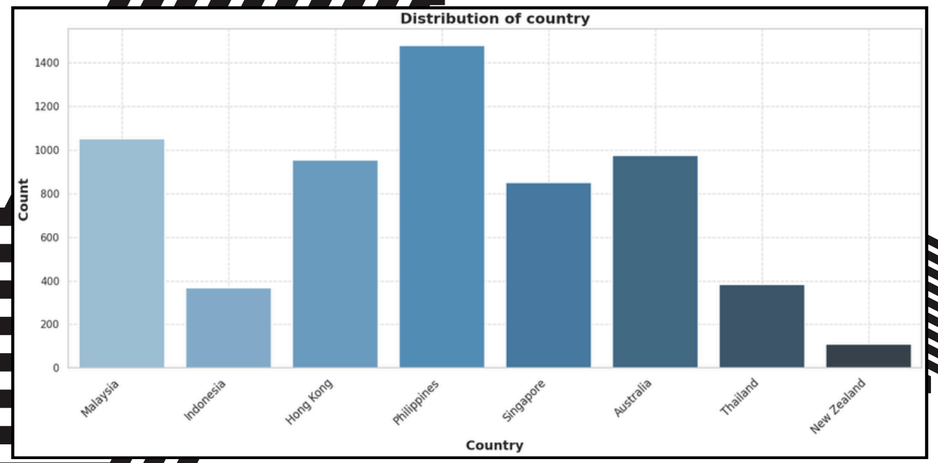
CATEGORICAL FEATURE INSIGHTS

Data Imbalances:
Over-representation of entry-level jobs in "level" feature suggests potentially skewed predictions toward over-represented groups. This also highlights the importance of certain features as a predictor.

Encoding Considerations:
Selection of encoding methods for categorical feature impacts model accuracy and interpretability.

Model Complexity:
High number of unique categories adds to model complexity.

Feature Engineering:
Combining visualizations with domain knowledge ensures accurate and unbiased salary predictions.

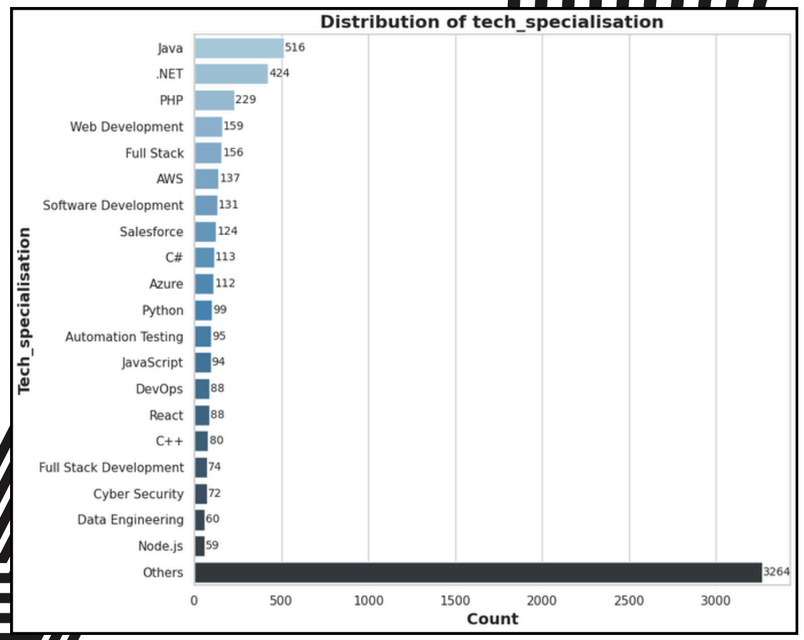


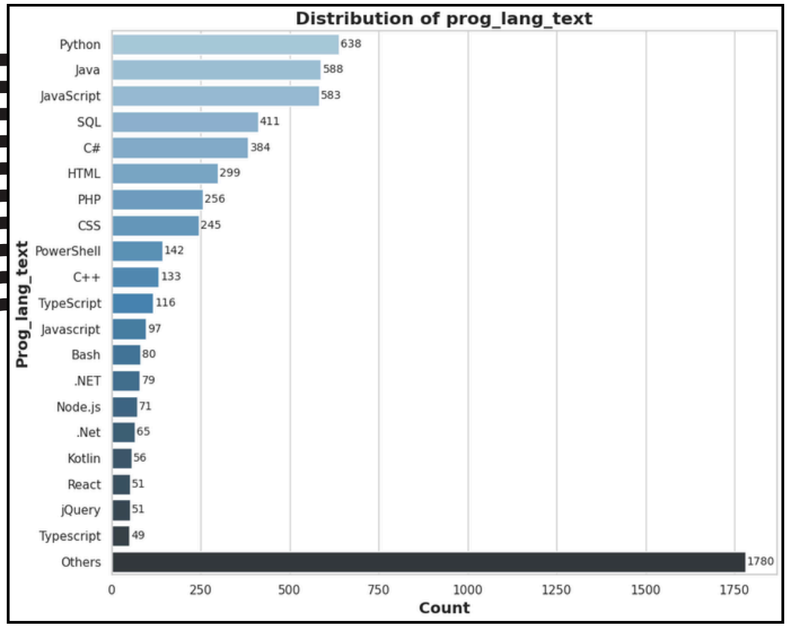
DATA ANALYSIS PROCESS

SPECIALIZATION AND PROGRAMMING LANGUAGE INSIGHTS

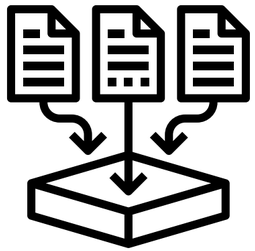
Data Imbalances:
Over-representation of certain specializations (e.g., "other") and programming languages (e.g., Python) potential introducing bias in salary predictions favoring these frequently represented areas.

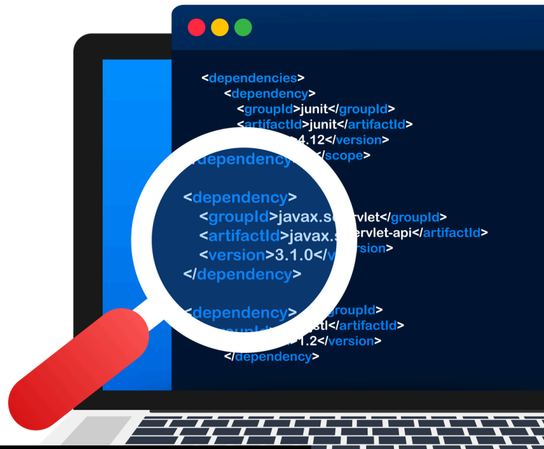
Mitigation Strategies:
Addressing these imbalances is crucial for unbiased and accurate predictions across all tech specializations and programming languages.





DATA ANALYSIS PROCESS





CORRELATION ANALYSIS INSIGHTS

Purpose:

Identifies linear relationships between numerical features and the target variable **(salary_from_usd)**. Additionally, guides feature selection by highlighting impactful predictors and reducing dataset noise.

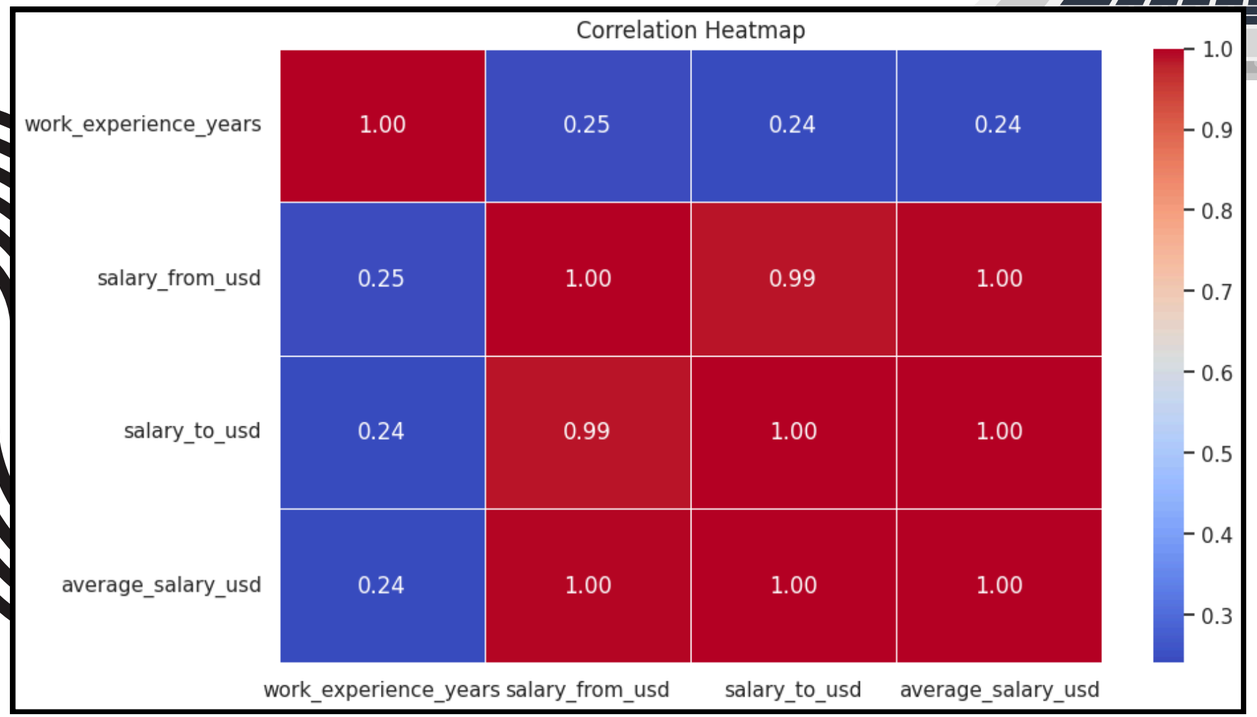
Key Observations:

Negative Correlation: Unexpected negative correlation between work experience and salary, requiring further investigation.

Redundancy: High correlation between salary_from_usd and salary_to_usd indicates potential multicollinearity issues.

Model Implications:

Negative trends might require feature engineering or algorithms handling non-linear relationships. Handling redundancy improves model reliability and accuracy.



DATA ANALYSIS PROCESS

GROUPING ANALYSIS AND KEY INSIGHTS

Objective:

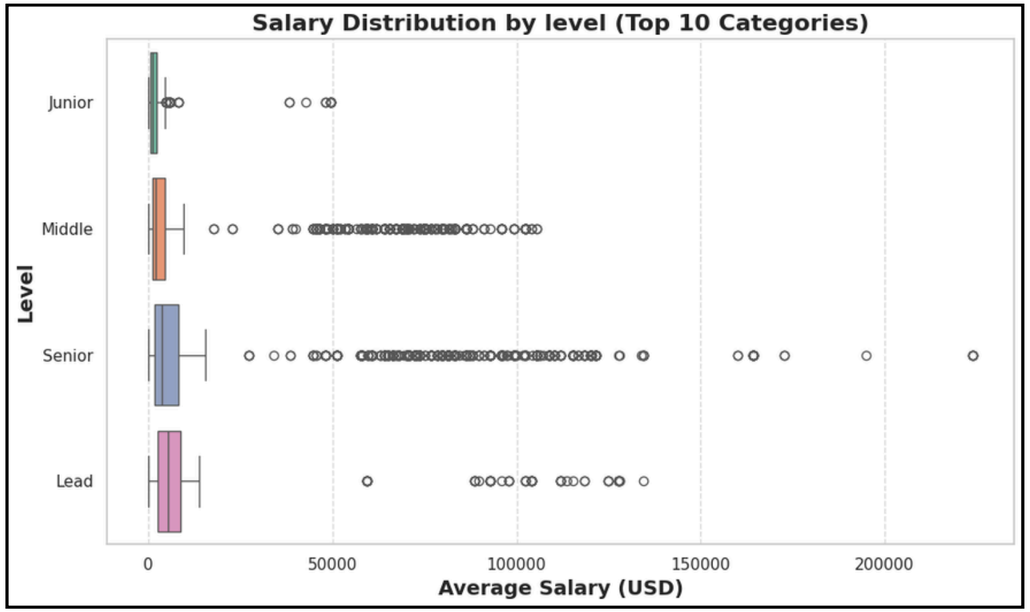
Examine how categorical features **(e.g., "level," "tech_specialisation," "education_level")** influence salary through averages and visualizations.

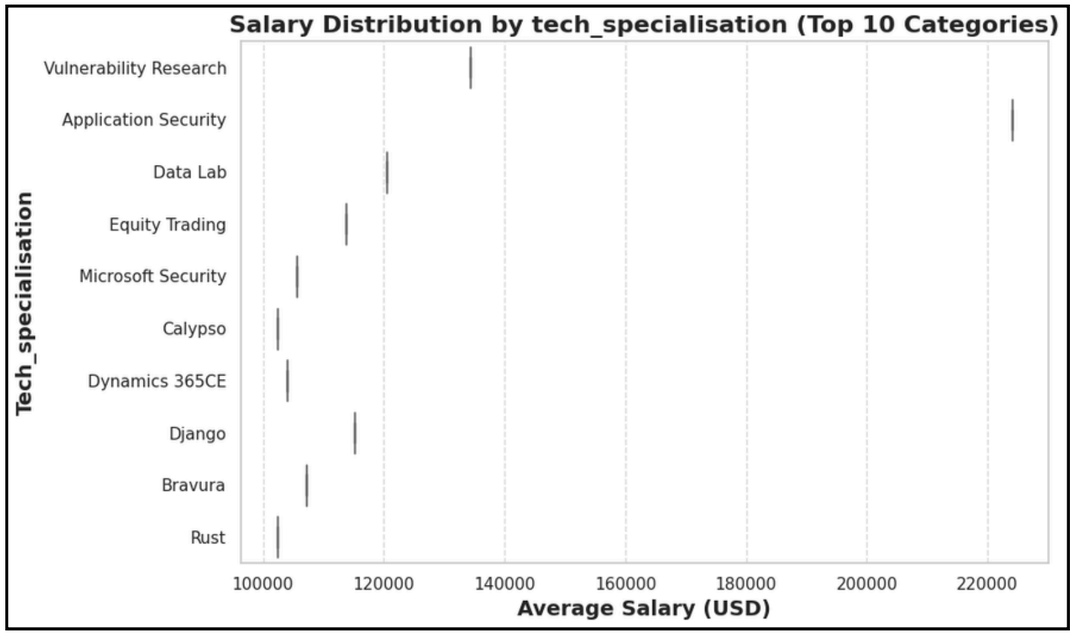
Key Observations:

- Median salary differences across categories highlight the importance of features for salary prediction.
- Clear trends, such as increasing median salary with higher job levels, underscore feature relevance **(e.g., junior vs. senior roles)**.
- Variability within categories (box length) reflects salary spread, informing model complexity.
- Outliers suggest unusual data points requiring further investigation.

Impact:

Insights guide feature engineering and model-building, ensuring accurate representation of categorical features.





DATA ANALYSIS PROCESS

GROUPING ANALYSIS AND KEY INSIGHTS

Objective:

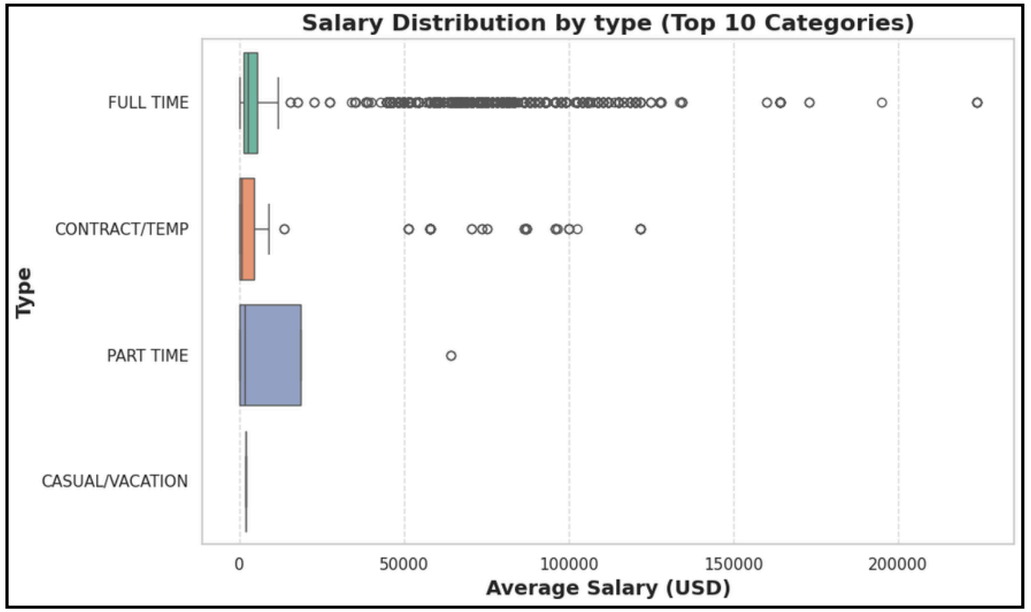
Examine how categorical features **(e.g., "level," "tech_specialisation," "education_level")** influence salary through averages and visualizations.

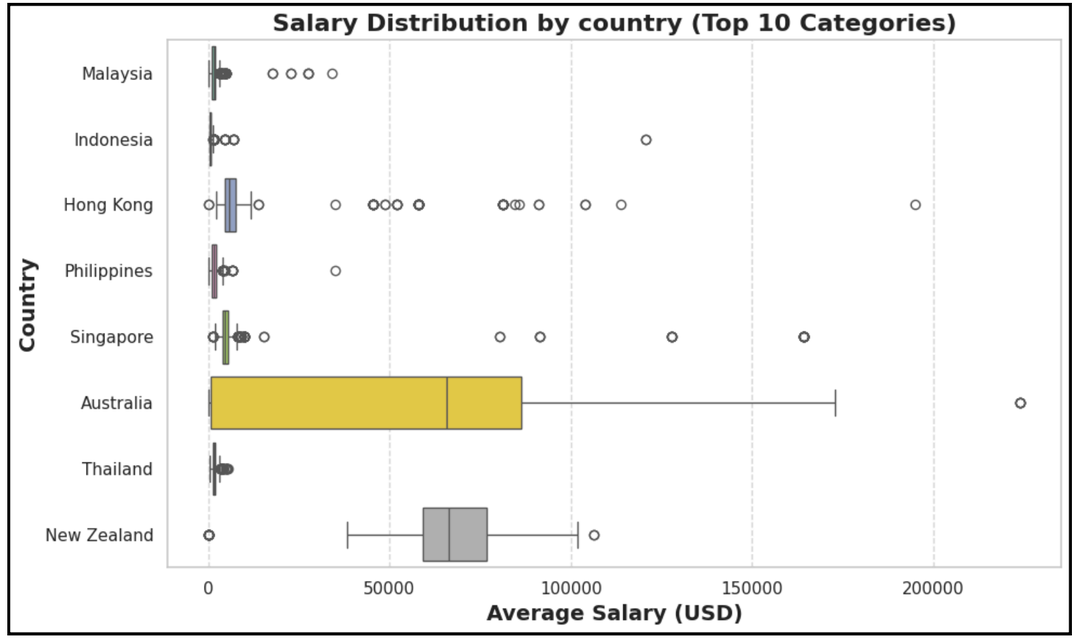
Key Observations:

- Median salary differences across categories highlight the importance of features for salary prediction.
- Clear trends, such as increasing median salary with higher job levels, underscore feature relevance **(e.g., junior vs. senior roles)**.
- Variability within categories (box length) reflects salary spread, informing model complexity.
- Outliers suggest unusual data points requiring further investigation.

Impact:

Insights guide feature engineering and model-building, ensuring accurate representation of categorical features.





DATA ANALYSIS PROCESS

GROUPING ANALYSIS AND KEY INSIGHTS

Objective:

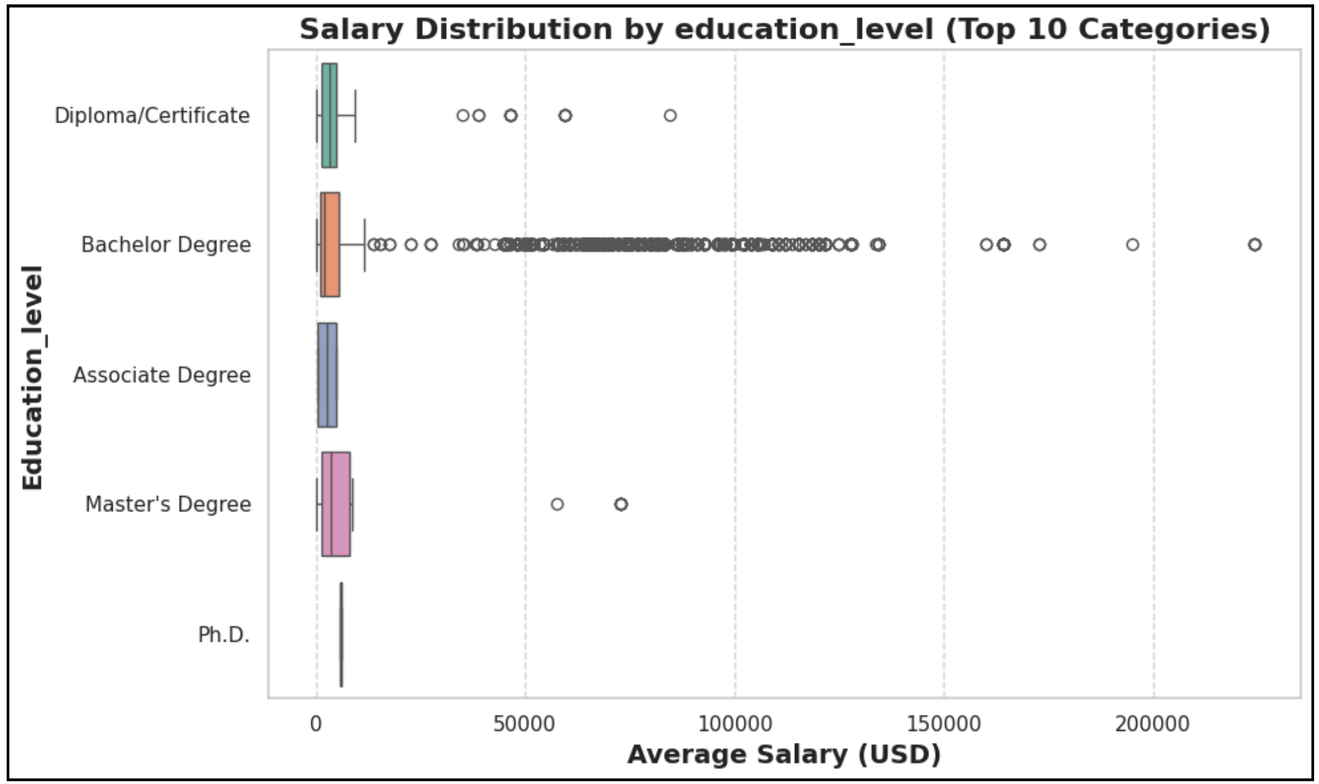
Examine how categorical features **(e.g., "level," "tech_specialisation," "education_level")** influence salary through averages and visualizations.

Key Observations:

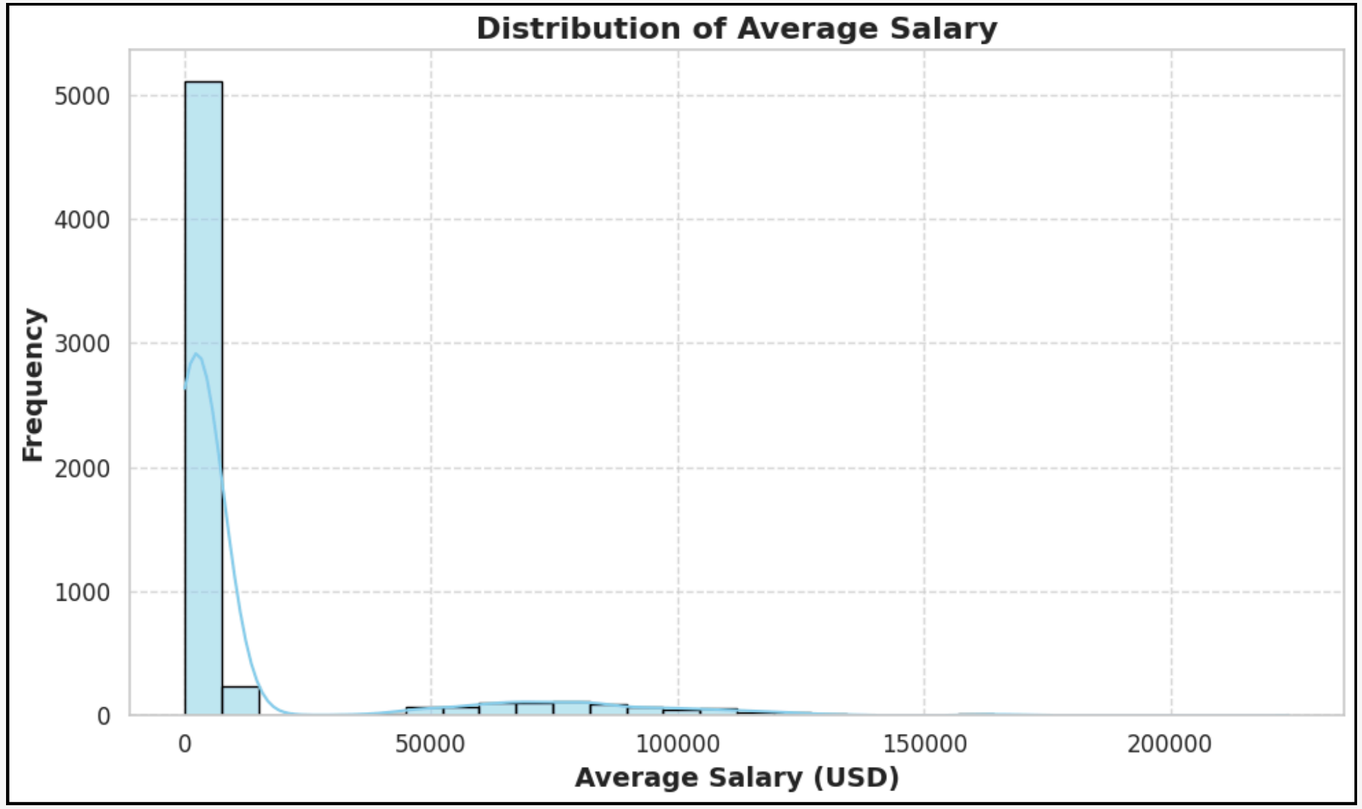
- Median salary differences across categories highlight the importance of features for salary prediction.
- Clear trends, such as increasing median salary with higher job levels, underscore feature relevance **(e.g., junior vs. senior roles)**.
- Variability within categories (box length) reflects salary spread, informing model complexity.
- Outliers suggest unusual data points requiring further investigation.

Impact:

Insights guide feature engineering and model-building, ensuring accurate representation of categorical features.



```python
# Set Seaborn style for better aesthetics
sns.set(style="whitegrid")

# Create the histogram with KDE
plt.figure(figsize=(10, 6))  # Set the figure size for better clarity
sns.histplot(final_data['average_salary_usd'], kde=True, color="skyblue", edgecolor="black", bins=30)

# Add title and labels
plt.title('Distribution of Average Salary', fontsize=16, fontweight='bold')
plt.xlabel('Average Salary (USD)', fontsize=14, fontweight='bold')
plt.ylabel('Frequency', fontsize=14, fontweight='bold')

# Customize ticks
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)

# Add gridlines for better readability
plt.grid(True, linestyle='--', alpha=0.7)

# Show the plot
plt.tight_layout()  # Ensure proper layout to prevent overlap
plt.show()
```



DATA VISUALIZATION

```python
# Set Seaborn style for better aesthetics
sns.set(style="whitegrid")

# Create the figure with size adjustments
plt.figure(figsize=(10, 6))

# Use seaborn's scatterplot with enhancements
sns.scatterplot(x='work_experience_years', y='salary_from_usd', data=final_data,
                color='skyblue', s=100, edgecolor='black', alpha=0.7)

# Title and labels with better aesthetics
plt.title('Work Experience vs. Salary', fontsize=18, fontweight='bold')
plt.xlabel('Work Experience (Years)', fontsize=14, fontweight='bold')
plt.ylabel('Salary (USD)', fontsize=14, fontweight='bold')

# Customize tick labels
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)

# Add gridlines for better readability
plt.grid(True, linestyle='--', linewidth=0.5, alpha=0.7)

# Show the plot with tight layout to prevent overlap
plt.tight_layout()
plt.show()
```

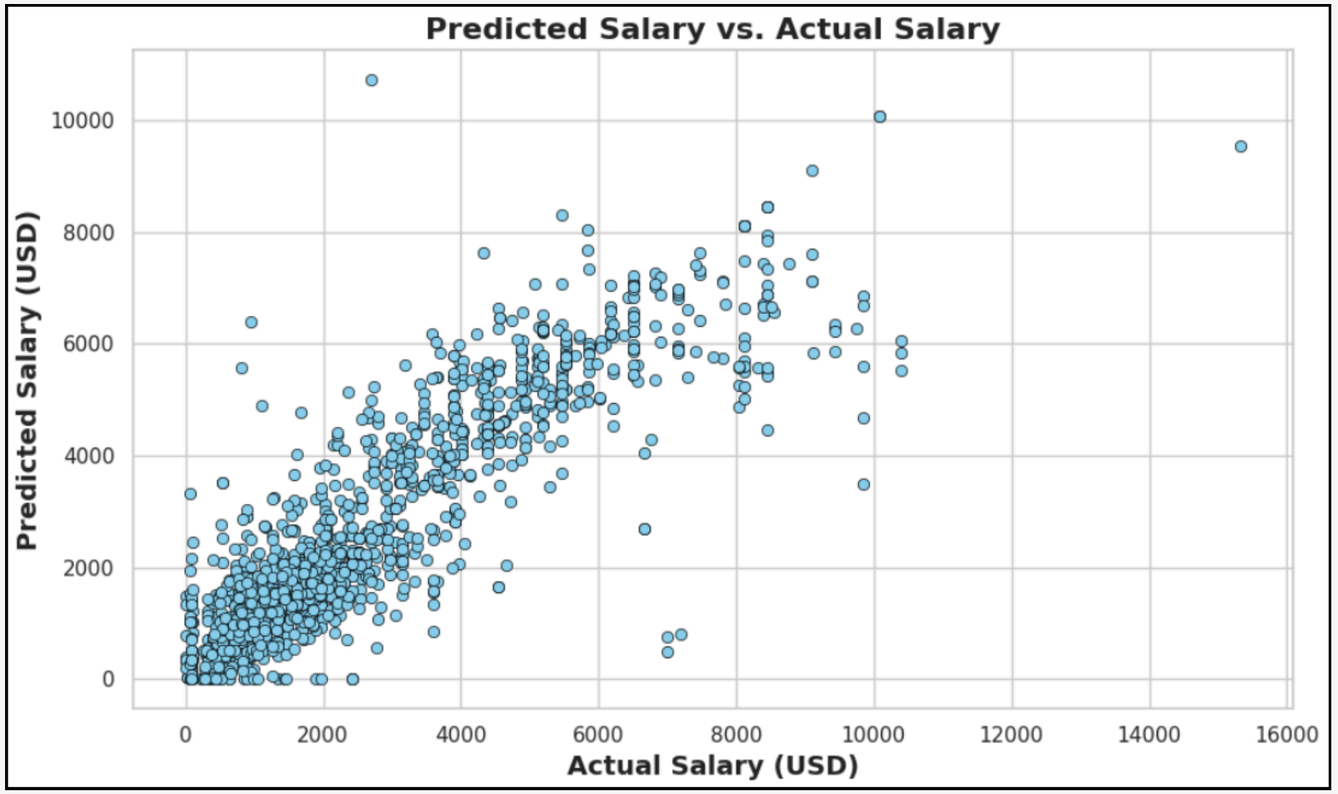
DATA VISUALIZATION

CHOSEN MODEL	MODEL EVALUATION	PREDICTION RESULTS
Supervised Model - Linear Regression	**Mean Squared Error:** 1065685.0590061855 1 065 685 **R-squared:** 0.7804080255795491 78% **Accuracy:** 0.8551810237203495 85% **Precision:** 0.848780487804878 85% **Recall:** 0.8656716417910447 87% **F1 Score:** 0.8571428571428571 86%	**Predicted Average Salary (USD):** 892.53 **Actual Average Salary (USD):** 892.50

MACHINE LEARNING MODEL

Visualizing the correlation between predicted and actual salaries, highlighting model accuracy and deviations for higher salary values.

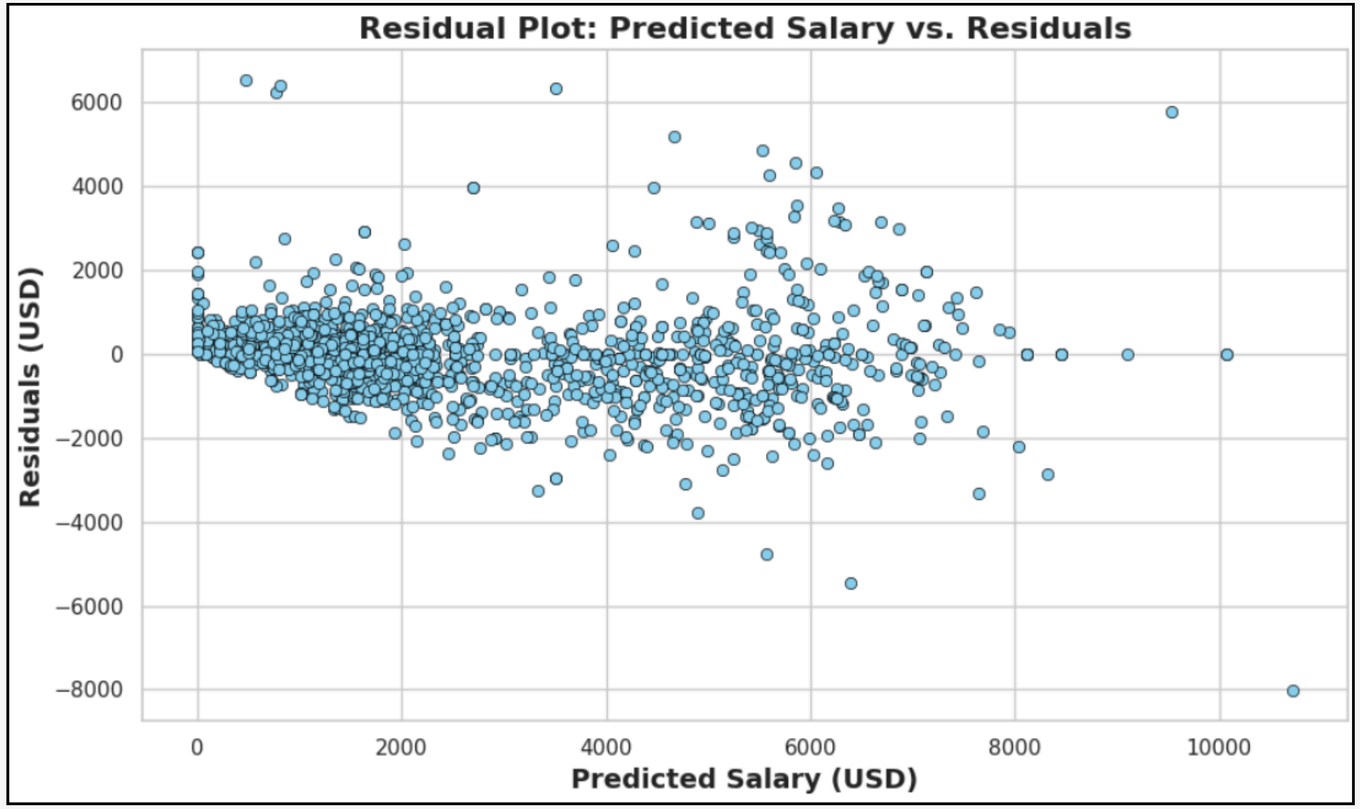
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=y_test, y=y_pred, color='skyblue', edgecolor='black')

# Title and labels
plt.title('Predicted Salary vs. Actual Salary', fontsize=16, fontweight='bold')
plt.xlabel('Actual Salary (USD)', fontsize=14, fontweight='bold')
plt.ylabel('Predicted Salary (USD)', fontsize=14, fontweight='bold')

# Show the plot
plt.tight_layout()
plt.show()
```



RESULTS & VISUALIZATION

Illustrating the residuals to assess model performance, showing patterns of deviation, especially for higher predicted salaries

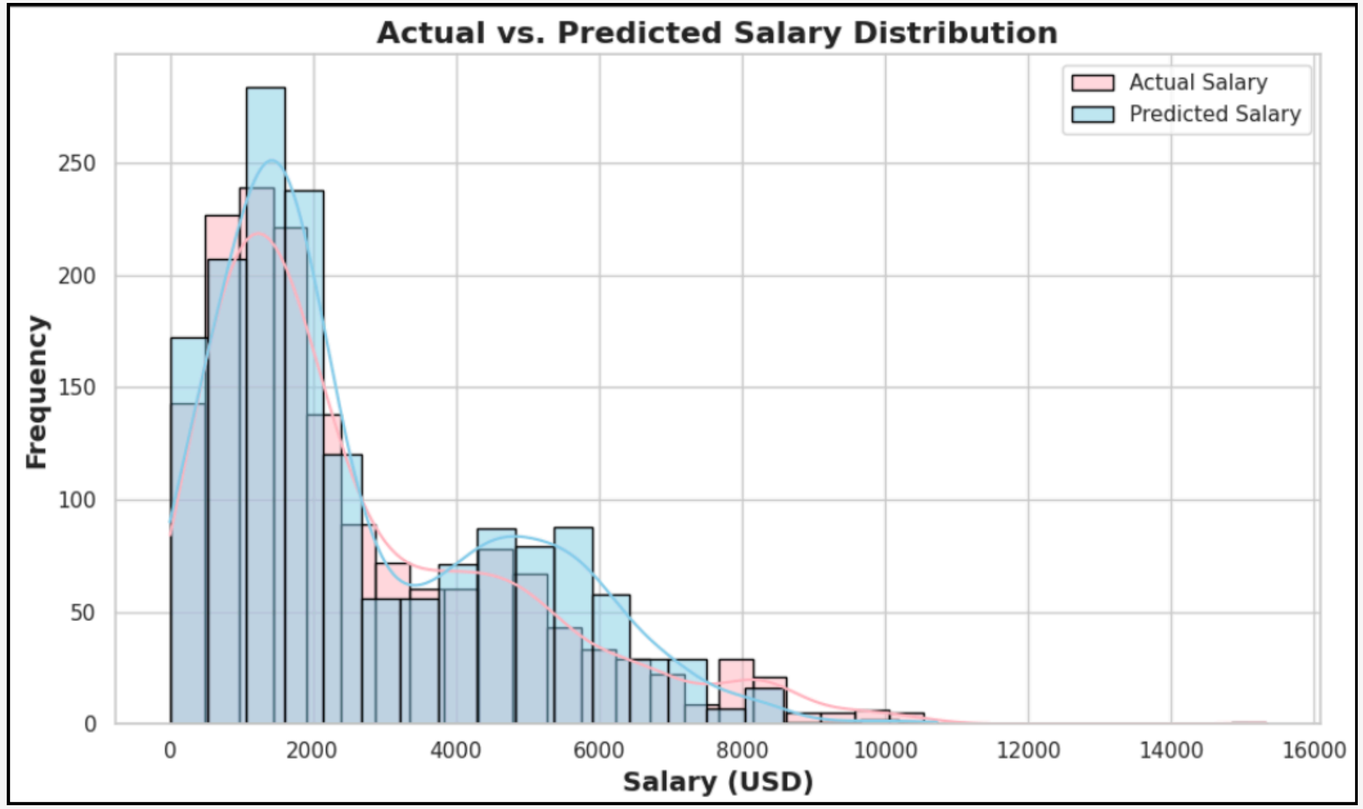
```python
# Calculate residuals
residuals = y_test - y_pred

plt.figure(figsize=(10, 6))
sns.scatterplot(x=y_pred, y=residuals, color='skyblue', edgecolor='black')

# Title and labels
plt.title('Residual Plot: Predicted Salary vs. Residuals', fontsize=16, fontweight='bold')
plt.xlabel('Predicted Salary (USD)', fontsize=14, fontweight='bold')
plt.ylabel('Residuals (USD)', fontsize=14, fontweight='bold')

# Show the plot
plt.tight_layout()
plt.show()
```



RESULTS & VISUALIZATION

Compares the distributions of predicted and actual salaries, demonstrating how closely the model's predictions match the actual data

```
plt.figure(figsize=(10, 6))
sns.histplot(y_test, color='lightpink', kde=True, label='Actual Salary', edgecolor='black')
sns.histplot(y_pred, color='skyblue', kde=True, label='Predicted Salary', edgecolor='black')

# Title and labels
plt.title('Actual vs. Predicted Salary Distribution', fontsize=16, fontweight='bold')
plt.xlabel('Salary (USD)', fontsize=14, fontweight='bold')
plt.ylabel('Frequency', fontsize=14, fontweight='bold')
plt.legend()

# Show the plot
plt.tight_layout()
plt.show()
```



RESULTS & VISUALIZATION

The students concluded that:

- The model shows reasonable performance with an R-squared of 0.78 and MSE of 1023527.37, explaining 85.51% of salary variance, which is a solid starting point for predicting IT professionals' salaries. Metrics like accuracy and F1 score suggest the model is decent at predicting salaries, though there's room for improvement. The precision and recall balance indicate that while the model identifies the most relevant salary predictions, it could still be more accurate in its predictions.
- Predicted salaries closely match actual salaries, showing good individual prediction accuracy.
- The above average performance could be attributed to factors like better data quality, effective feature engineering, and model tuning.

CONCLUSIONS & RECOMENDATIONS

In terms of real-world implications:

- By providing more accurate salary predictions based on factors such as work experience, education, and skill set, the model can help individuals make informed career decisions, negotiate salaries, and set realistic expectations.
- Organizations, on the other hand, can use the model to ensure competitive and equitable salary offers, attract and retain talent, and manage compensation strategies more effectively.
- However, the model's current performance suggests that there is still room for refinement.

CONCLUSIONS & RECOMENDATIONS

Based on the results, the students recommend that:

1. Feature Engineering
2. Model Complexity
3. Data Quality

CONCLUSIONS & RECOMENDATIONS

PRESENTED BY:
GROUP 5 4CSC
GUEVARRA
JAVIER
LUCES
ROMANES
TAPAO

DATA ANALYSIS AND VISUALIZATION
14 DECEMBER 2024

SALARY DECODED

A Data-Driven Analysis of IT Compensation