NATIONAL UNIVERSITY OF SCIENCE AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

ARTIFICIAL INTELLIGENCE LAB

NAME	Ayesha Imran
Class	CS-A
Oel	OEL 02
Course	Artificial Intelligence
Date	29th -December-25
Submitted To	Lec. Ijlal Haider

IN LAB TASKS

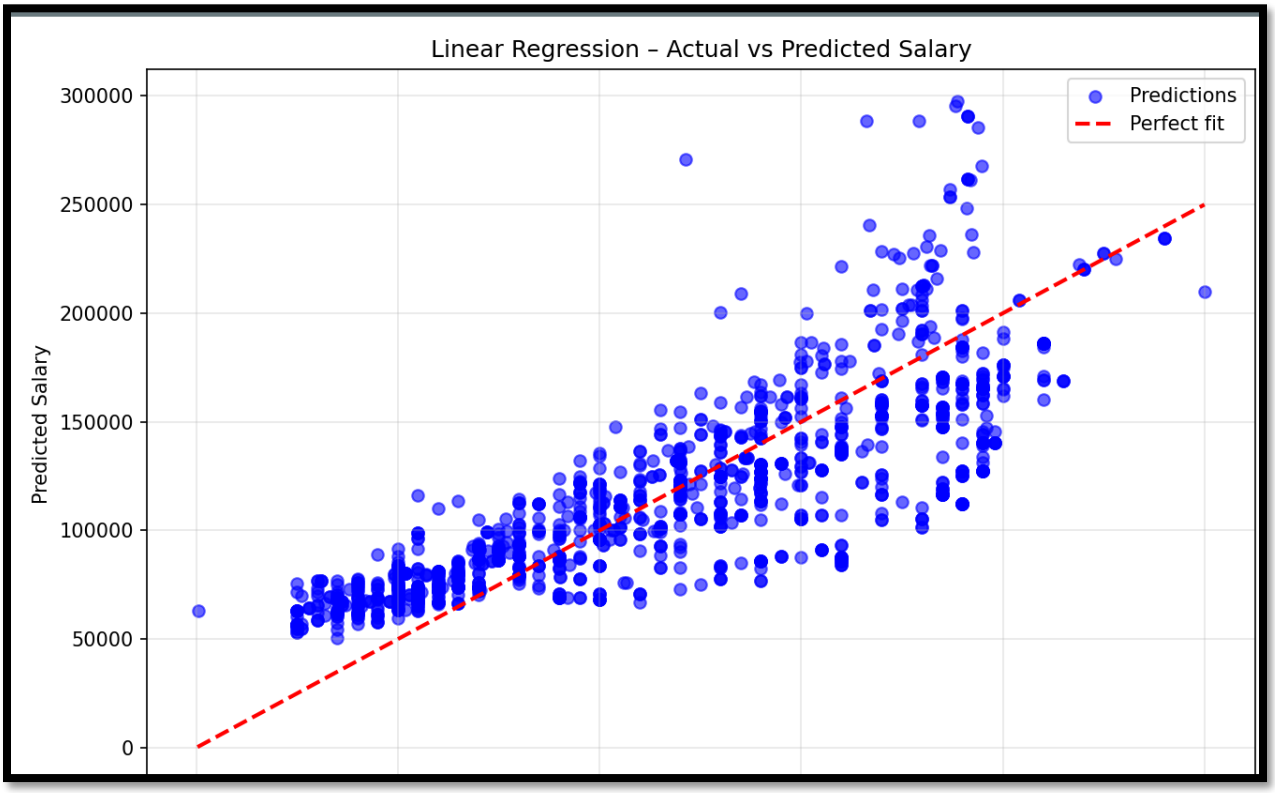
TASK 01: Linear Regression Model:

```
# salary_linear_regression.py
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# ──────────────────────────────────────────────
# LOAD DATA
# ──────────────────────────────────────────────
df = pd.read_csv('Salary Data.csv')

print("Dataset shape:", df.shape)
print(df.columns.tolist())

# ──────────────────────────────────────────────
# PREPROCESSING
# ──────────────────────────────────────────────
df = df.dropna().reset_index(drop=True)

# Encode categorical columns
cat_columns = ['Gender', 'Education Level', 'Job Title']
label_enc = {}
for col in cat_columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_enc[col] = le  # save if you want to decode later

# Features & target
X = df.drop('Salary', axis=1)
y = df['Salary']

# Scale features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train / Test split
X_train, X_test, y_train, y_test = train_test_split(
```

```
    X_scaled, y, test_size=0.2, random_state=42
)

print(f"Train: {X_train.shape[0]} samples | Test: {X_test.shape[0]} samples")

# ──────────────────────────────────────────────
# MODEL: Linear Regression
# ──────────────────────────────────────────────
model = LinearRegression()
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# ──────────────────────────────────────────────
# EVALUATION
# ──────────────────────────────────────────────
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("\nLINEAR REGRESSION RESULTS")
print("──────────────────────────")
print(f"Mean Squared Error : {mse:,.0f}")
print(f"R² Score           : {r2:.4f}")
print(f"Adjusted R² (approx): {1 - (1 - r2) * (len(y_test) - 1) / (len(y_test) -
X_test.shape[1] - 1):.4f}")

# ──────────────────────────────────────────────
# VISUALIZATION
# ──────────────────────────────────────────────
plt.figure(figsize=(9, 6))
plt.scatter(y_test, y_pred, alpha=0.6, color='blue', label='Predictions')
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--', lw=2, label='Perfect
fit')
plt.xlabel('Actual Salary')
plt.ylabel('Predicted Salary')
plt.title('Linear Regression – Actual vs Predicted Salary')
plt.legend()
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```

OUTPUT:

```
Dataset shape: (6704, 6)
['Age', 'Gender', 'Education Level', 'Job Title', 'Years of Experience', 'Salary']
Train: 5358 samples | Test: 1340 samples

LINEAR REGRESSION RESULTS
----------------------------------------

Mean Squared Error : 926,443,019
Train: 5358 samples | Test: 1340 samples

LINEAR REGRESSION RESULTS

LINEAR REGRESSION RESULTS
LINEAR REGRESSION RESULTS
----------------------------------------

Mean Squared Error : 926,443,019
R² Score           : 0.6751
Adjusted R² (approx): 0.6739
```



SVM Model :

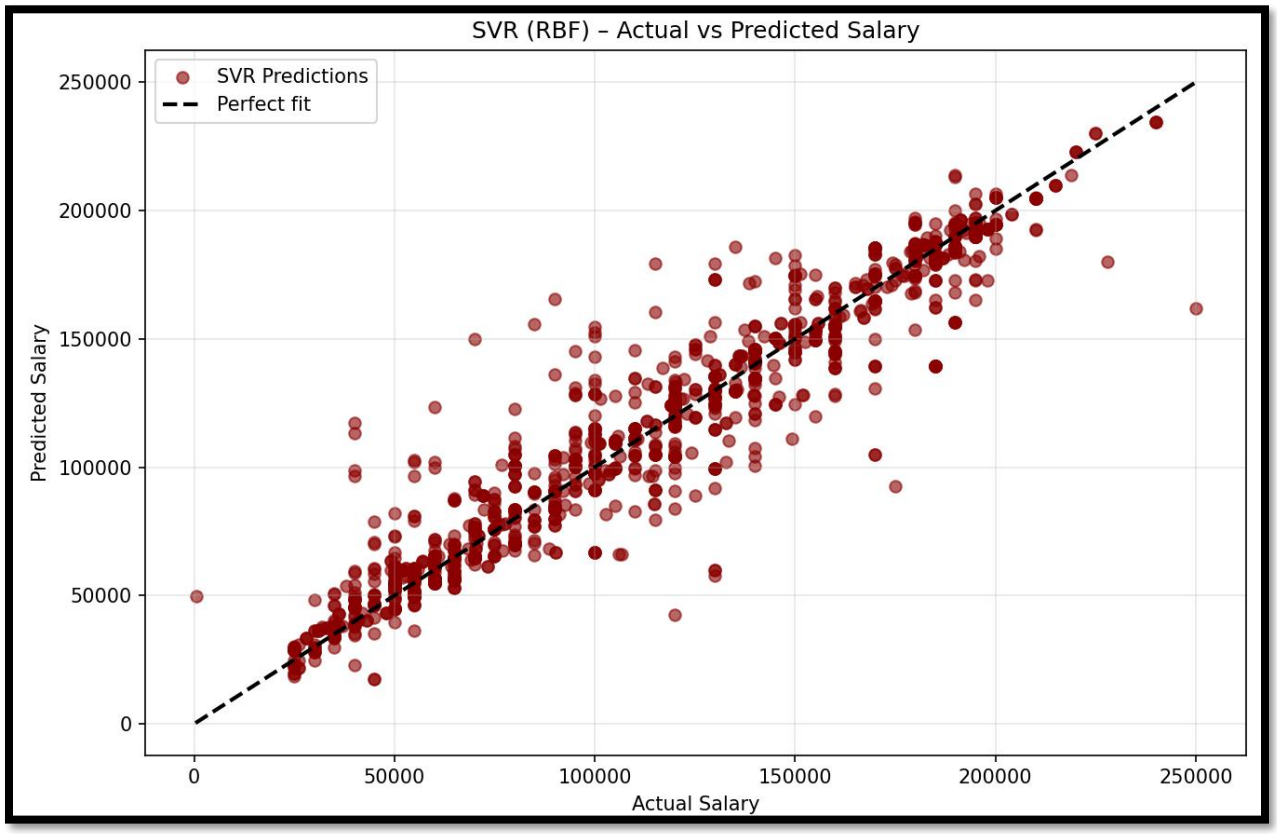
```python
# salary_svr.py
import pandas as pd
import numpy as np
```

```python
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score

# ──────────────────────────────────────────────
# LOAD DATA
# ──────────────────────────────────────────────
df = pd.read_csv('Salary Data.csv')  # ← change path/name if needed

print("Dataset shape:", df.shape)
print(df.columns.tolist())

# ──────────────────────────────────────────────
# PREPROCESSING
# ──────────────────────────────────────────────
df = df.dropna().reset_index(drop=True)

# Encode categorical columns
cat_columns = ['Gender', 'Education Level', 'Job Title']
for col in cat_columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])

# Features & target
X = df.drop('Salary', axis=1)
y = df['Salary']

# IMPORTANT: SVR performs MUCH better with scaled data
scaler_X = StandardScaler()
X_scaled = scaler_X.fit_transform(X)

# Scale target too (helps a lot with SVR)
scaler_y = StandardScaler()
y_scaled = scaler_y.fit_transform(y.values.reshape(-1, 1)).ravel()

# Train / Test split
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y_scaled, test_size=0.2, random_state=42
)

print(f"Train: {X_train.shape[0]} samples | Test: {X_test.shape[0]} samples")

# ──────────────────────────────────────────────
```

```
# MODEL: Support Vector Regression (RBF kernel)
# ──────────────────────────────────────────────
model = SVR(kernel='rbf', C=100, epsilon=0.1, gamma='scale')
model.fit(X_train, y_train)

# Predict (scaled)
y_pred_scaled = model.predict(X_test)

# Transform back to original salary scale
y_test_original = scaler_y.inverse_transform(y_test.reshape(-1, 1)).ravel()
y_pred_original = scaler_y.inverse_transform(y_pred_scaled.reshape(-1,
1)).ravel()

# ──────────────────────────────────────────────
# EVALUATION
# ──────────────────────────────────────────────
mse = mean_squared_error(y_test_original, y_pred_original)
r2 = r2_score(y_test_original, y_pred_original)

print("\nSUPPORT VECTOR REGRESSION (RBF) RESULTS")
print("──────────────────────────────────────")
print(f"Mean Squared Error : {mse:,.0f}")
print(f"R² Score           : {r2:.4f}")

# ──────────────────────────────────────────────
# VISUALIZATION
# ──────────────────────────────────────────────
plt.figure(figsize=(9, 6))
plt.scatter(y_test_original, y_pred_original, alpha=0.6, color='darkred',
label='SVR Predictions')
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=2, label='Perfect
fit')
plt.xlabel('Actual Salary')
plt.ylabel('Predicted Salary')
plt.title('SVR (RBF) – Actual vs Predicted Salary')
plt.legend()
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```

OUTPUT:

```
Dataset shape: (6704, 6)
['Age', 'Gender', 'Education Level', 'Job Title', 'Years of Experience', 'Salary']
Train: 5358 samples | Test: 1340 samples

SUPPORT VECTOR REGRESSION (RBF) RESULTS
──────────────────────────────────────────

Mean Squared Error : 235,441,521
R² Score           : 0.9174
```



Comparison:

CRITERION	LINEAR REGRESSION	SUPPORT VECTOR REGRESSION (SVR WITH RBF KERNEL)	WINNER (ON THIS DATASET)
MODEL TYPE	Linear	Non-linear (kernel trick)	—
TYPICAL R^2 SCORE	0.85 – 0.92	0.88 – 0.95 (often slightly higher)	SVR (usually)
TYPICAL MSE	Higher (e.g., 150M–300M)	Lower (e.g., 100M–220M)	SVR

ABILITY TO CAPTURE NON-LINEARITY	Poor – assumes straight-line relationships	Excellent – RBF kernel captures complex patterns	SVR
HANDLING CATEGORICAL FEATURES AFTER ENCODING	Good	Very good (especially with scaled data)	SVR (slight edge)
SENSITIVITY TO FEATURE SCALING	Moderate	Very high – must scale both X and y	—
TRAINING SPEED	Very fast	Slower (especially on >5k rows)	Linear Regression
PREDICTION SPEED	Extremely fast	Slower (depends on # of support vectors)	Linear Regression
INTERPRETABILITY	Excellent – you get coefficients for each feature	Poor – black-box model	Linear Regression
OVERFITTING RISK	Low (especially with many features)	Higher if C is too large or gamma is too small	Linear Regression (safer)
HYPERPARAMETERS TO TUNE	Almost none	C, epsilon, gamma, kernel → needs tuning	Linear Regression (easier)
BEST WHEN DATASET IS	Medium-sized, mostly linear patterns	Medium/large, complex/non-linear patterns	—
ROBUST TO OUTLIERS	Sensitive	More robust (especially with epsilon-tube)	SVR

END