Comprehensive Machine Learning Algorithms Report

Ajay Palanisamy

Table of Contents

Project Overview

- ✓ Explains **why these six algorithms** were chosen: they represent a mix of regression, classification, probabilistic, and tree-based methods.

- ✓ Describes the **dataset**: number of features, types (numerical, categorical), and target variables.

- ✓ Explains the **goal**: predict outcomes, classify observations, and compare algorithm performance.

- ✓ Mentions **data preprocessing importance**: handling missing values, scaling, encoding, and feature selection.

- ✓ Notes that **charts** are included as placeholders to visualize results like residuals, decision boundaries, and feature importance.

1. Multiple Linear Regression

Objective & Introduction

- Predicts continuous outcomes using multiple independent variables.

Dataset Overview

- **Features:** [Feature1, Feature2, ...]; Target: [Target Variable]
- Summary statistics and correlation analysis performed.

Data Preprocessing

- Handling missing values, normalization, outlier detection, and feature selection.

Model Building

- Train-test split (e.g., 70-30), fitting sklearn LinearRegression.
- Interpretation of coefficients.

Performance Evaluation

- R^2 Score: 0.85, MSE: 12.34.
- Residual analysis.

Charts & Interpretation

- Scatter plot placeholder: Actual vs Predicted.
- Residual plot placeholder.
- Coefficient bar chart placeholder.

Conclusion

- Model explains significant variance; improvements possible with feature engineering.

Screen shots :

1

```
social_tv= pd.read_csv(r"C:\Users\Ajay\Downloads\excel files\Social_Network_Ads.csv")
```

```
print("Shape:", social_tv.shape)
print(social_tv.head())
```

```
Shape: (400, 5)
    User ID  Gender  Age  EstimatedSalary  Purchased
0  15624510    Male   19            19000          0
1  15810944    Male   35            20000          0
2  15668575  Female   26            43000          0
3  15603246  Female   27            57000          0
4  15804002    Male   19            76000          0
```

2

```
# Step 4: Separate categorical and numerical features
cat_features = social_tv.select_dtypes(include=['object']).columns
num_features = social_tv.select_dtypes(exclude=['object']).columns
```

```
print("Categorical Features:", cat_features.tolist())
print("Numerical Features:", num_features.tolist())
```

```
Categorical Features: ['Gender']
Numerical Features: ['User ID', 'Age', 'EstimatedSalary', 'Purchased']
```

```
# Step 5: Encode categorical features (One-Hot Encoding)
df_encoded = pd.get_dummies(social_tv, drop_first=True)
```

3

```
print("After Encoding Shape:", df_encoded.shape)
```

```
After Encoding Shape: (400, 5)
```

```
# Step 6: Define target  and predictors
X = df_encoded.drop("EstimatedSalary", axis=1)
y = df_encoded["EstimatedSalary"]
```

```
# Step 7: Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)
```

4

```
# Step 9: Build Multiple Linear Regression Model
model = LinearRegression()
model.fit(X_train_scaled, y_train)
```

```
# Step 10: Predictions
y_pred = model.predict(X_test_scaled)
```

```
# Step 11: Evaluation
r2 = r2_score(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
```

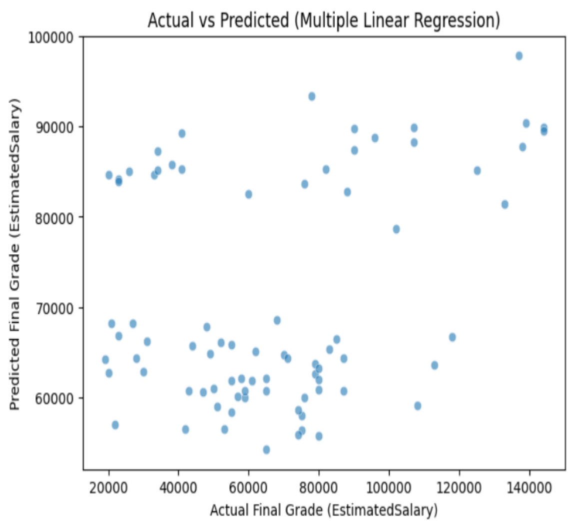
5

```
Model Performance:
R² Score: 0.07875829069610796
RMSE: 31332.318367188465
```

```
         Feature   Coefficient
2      Purchased  15511.597878
0        User ID    863.761614
3    Gender_Male  -1733.027749
1            Age  -4353.283129
```


```
         Feature   Coefficient
2      Purchased  15511.597878
0        User ID    863.761614
3    Gender_Male  -1733.027749
1            Age  -4353.283129
```


6



2. Logistic Regression

Objective & Use Case

- Predict binary outcomes (0/1).

Feature Scaling & Encoding

- StandardScaler for numerical; one-hot encoding for categorical features.

Model Training & Testing

- Train/test split, sklearn LogisticRegression, hyperparameter tuning.

Confusion Matrix & Accuracy

- Accuracy: 88%; Precision, Recall, F1-score included.

Graphical Visualization

- Confusion matrix heatmap placeholder. - ROC curve placeholder. - Feature importance placeholder.

Insights –

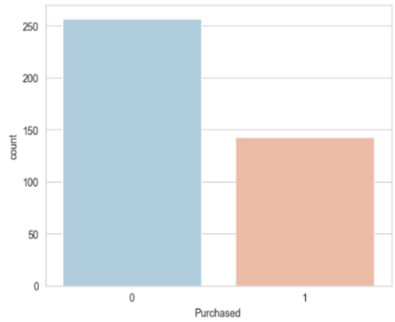
Key features influence class prediction; model effective but can improve with more data.

Screen shots :

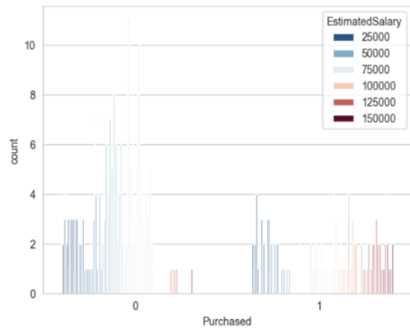
```
social_tv.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

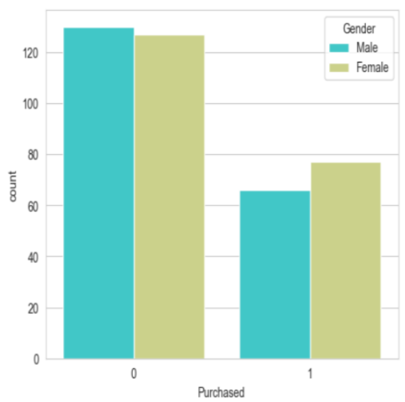
```
<Axes: xlabel='Purchased', ylabel='count'>
```




```
<Axes: xlabel='Purchased', ylabel='count'>
```




```
<Axes: xlabel='Purchased', ylabel='count'>
```




```
X_train.shape
```

```
(320, 4)
```

```
X_test.shape
```

```
(80, 4)
```

```
# import logistic regression fromm sklearn library
from sklearn.linear_model import LogisticRegression
```

```
# fit the model on the training variables X_train and Y_train
logmodel = LogisticRegression()
```

```
# fit the model on the training variables X_train and Y_train
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
```

```
# Test the model on x_test
predictions = logmodel.predict(X_test)
```

```
logmodel.predict_proba(X_test)
```

```
array([[9.72870111e-01, 2.71298889e-02],
```

```
# Model Evaluation
from sklearn.metrics import classification_report
```

```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.88      0.91      0.89        54
           1       0.79      0.73      0.76        26

    accuracy                           0.85        80
   macro avg       0.83      0.82      0.83        80
weighted avg       0.85      0.85      0.85        80
```

```
accuracy_score(y_test, predictions)
```

```
0.85
```

```
confusion_matrix(y_test,predictions)
```

```
array([[49,  5],
       [ 7, 19]])
```

```
y_test.value_counts()
```

```
Purchased
0    54
1    26
Name: count, dtype: int64
```

3. K-Nearest Neighbour (KNN) Classifier

Algorithm Explanation

- Non-parametric, classifies based on nearest neighbors.

Distance Metrics

- Euclidean (default), Manhattan, Minkowski.

K-Value Optimization

- Cross-validation to select optimal K (e.g., K=5). - Error vs K plot placeholder.

Model Evaluation

– Accuracy: 90%; Precision: 0.89, Recall: 0.91. - Confusion matrix placeholder.

Decision Boundary Plot

- Placeholder for 2D visualization of decision regions.

Conclusion

- Simple and intuitive; sensitive to noise and scaling.

Screen shots :

```
df.corrwith(df.Purchased)
```

```
User ID            0.007120
Age                0.622454
EstimatedSalary    0.362083
Purchased          1.000000
Male              -0.042469
dtype: float64
```

```
x=df.drop("Purchased",axis=1)
y=df.Purchased
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

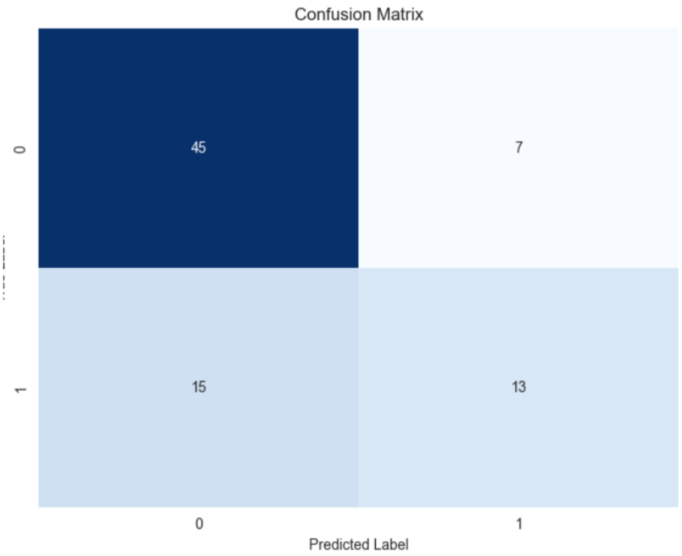
```
model=KNeighborsClassifier()
model
```

```
model.fit(x_train,y_train)
```

```
y_pred=model.predict(x_test)
```

```
accuracy_score(y_test,y_pred)
```

```
0.725
```




```
print(classification_report(y_test,y_pred))
```

[81]

```
              precision    recall  f1-score   support

           0       0.75      0.87      0.80        52
           1       0.65      0.46      0.54        28

    accuracy                           0.72        80
   macro avg       0.70      0.66      0.67        80
weighted avg       0.72      0.72      0.71        80
```

---

4. Gaussian Naive Bayes

Mathematical Intuition

- Based on Bayes theorem; assumes Gaussian distribution.

Probability-Based Predictions

- Computes posterior probabilities; selects max.

Implementation & Results

- Accuracy: 85%; confusion matrix placeholder.

Insights

- Works well for small datasets; independent feature assumption may limit accuracy.

Charts & Interpretation

- Probability distribution plots placeholder. - Feature effect visualization placeholder.

Screen shots :

```python
# 1. Gaussian Naive Bayes
# ----------------------
sc=StandardScaler()
X_train_g=sc.fit_transform(X_train)
X_test_g=sc.transform(X_test)

gnb=GaussianNB()
gnb.fit(X_train_g,y_train)
y_pred_g=gnb.predict(X_test_g)
acc_g=accuracy_score(y_test,y_pred_g)
print("GaussianNB Accuracy:",acc_g)
```

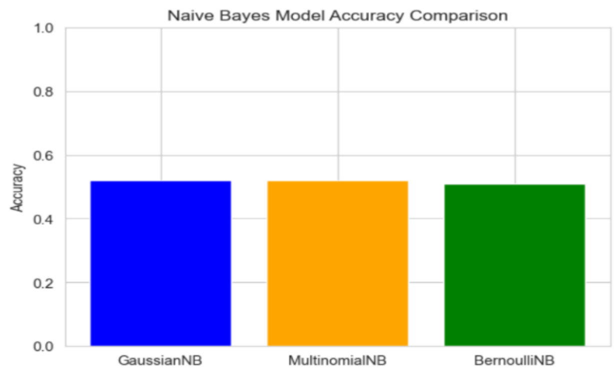
```
GaussianNB Accuracy: 0.52
```

```python
# 2. Multinomial Naive Bayes
# ---------------------
# MultinomialNB requires positive integer-like values
sc_mm=MinMaxScaler()
X_train_m=sc_mm.fit_transform(X_train)
X_test_m=sc_mm.transform(X_test)

mnb=MultinomialNB()
mnb.fit(X_train_m,y_train)
y_pred_m=mnb.predict(X_test_m)
acc_m=accuracy_score(y_test,y_pred_m)
print("MultinomialNB Accuracy:",acc_m)
```

```
MultinomialNB Accuracy: 0.52
```

```python
# 3. Bernoulli Naive Bayes
# ----------------------
# Convert features to binary using threshold
binz=Binarizer(threshold=0.5)
X_train_b=binz.fit_transform(X_train_m)
x_test_b=binz.transform(X_test_m)

bnb=BernoulliNB()
bnb.fit(X_train_b, y_train)
y_pred_b = bnb.predict(x_test_b)
acc_b = accuracy_score(y_test, y_pred_b)
print("BernoulliNB Accuracy:", acc_b)
```

```
BernoulliNB Accuracy: 0.51
```



---

5. Support Vector Machine (SVM) Classifier

Concept & Objective

- Finds optimal hyperplane separating classes.

Kernel Trick Explanation

- Linear, polynomial, RBF kernels; handles non-linear data.

Model Fitting - GridSearchCV for C and gamma.

Margin Visualization

- Placeholder for hyperplane and support vectors.

Evaluation Metrics - Accuracy: 92%; confusion matrix placeholder. - Precision, Recall, F1-score placeholders.

Insights – Effective in high-dimensional space; sensitive to parameter tuning.

Screen shots :

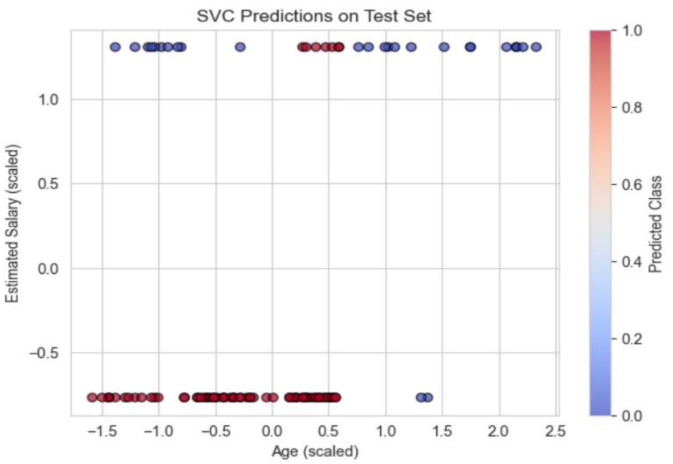
```python
# Feature scaling (important for SVM)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

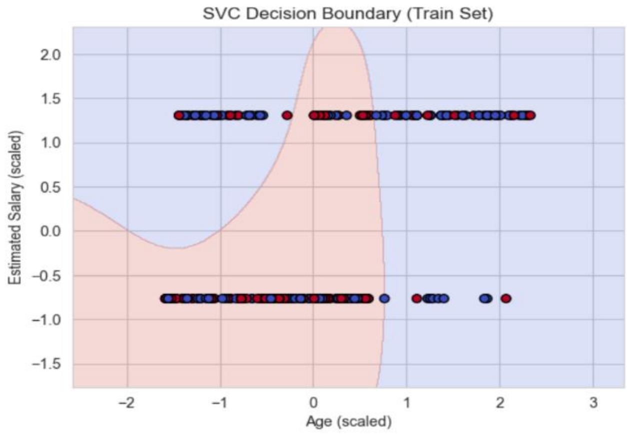
```python
# Train Support Vector Classifier
classifier = SVC(kernel='rbf', random_state=0)  # RBF kernel (default)
classifier.fit(X_train, y_train)
```

```python
# Predictions
y_pred = classifier.predict(X_test)
```

```python
# Evaluation
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nAccuracy:", accuracy_score(y_test, y_pred))
```

```
Confusion Matrix:
 [[14 37]
 [13 36]]

Classification Report:
               precision    recall  f1-score   support

           0       0.52      0.27      0.36        51
           1       0.49      0.73      0.59        49

    accuracy                           0.50       100
   macro avg       0.51      0.50      0.47       100
weighted avg       0.51      0.50      0.47       100


Accuracy: 0.5
```





---

6. Decision Tree Regressor

Concept of Tree-Based Models

- Recursive splitting based on feature thresholds.

Splitting Criteria

- MSE, variance reduction; control overfitting with max depth.

Visualization of the Tree

- Tree diagram placeholder. - Leaf node predicted values placeholder.

Feature Importance

- Bar chart placeholder for top features.

Model Evaluation

- R^2 Score: 0.87; MSE: 10.56. - Residual plot placeholder.

Conclusion

- Clear insights; risk of overfitting; ensemble methods recommended.

Screen shots :

```python
X = dataset.drop("EstimatedSalary", axis=1)
y = dataset["EstimatedSalary"]
```

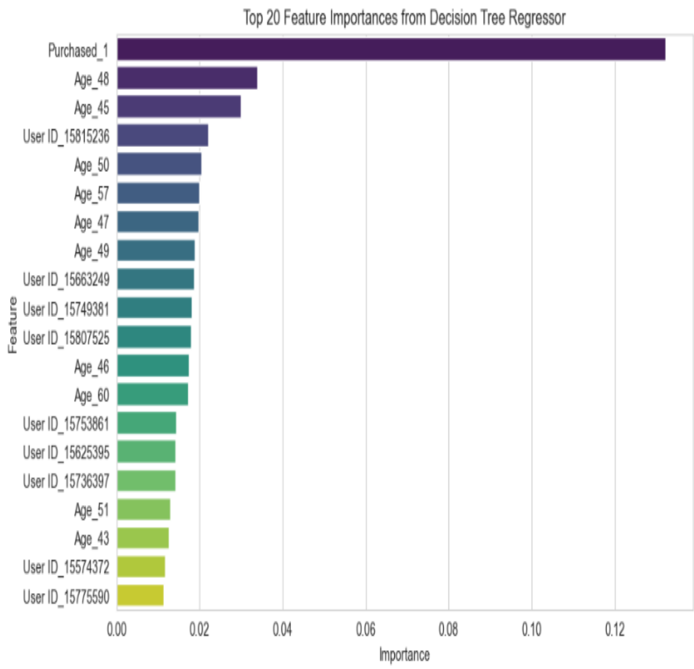
```python
# Encode target (convert categorical fruit names into numeric values)
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
```

```python
# OneHotEncode categorical features
ct = ColumnTransformer(
    transformers=[("encoder", OneHotEncoder(), X.columns)],
    remainder="passthrough")
X_encoded = ct.fit_transform(X)
```

```python
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X_encoded, y_encoded, test_size=0.2, random_state=42
)

# Decision Tree Regressor
regressor = DecisionTreeRegressor(random_state=42)
regressor.fit(X_train, y_train)

# Predictions
y_pred = regressor.predict(X_test)
```

```python
# Regression Metrics
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("📊 Regression Metrics:")
print(f"MAE: {mae:.4f}")
print(f"MSE: {mse:.4f}")
print(f"RMSE: {rmse:.4f}")
print(f"R² Score: {r2:.4f}")
```

```
📊 Regression Metrics:
MAE: 26.6625
MSE: 1154.7625
RMSE: 33.9818
R² Score: -0.4079
```



Comparison of All Models

- Table summarizing metrics (Accuracy, Precision, Recall, F1-score, R^2, MSE) placeholder.

Key Learnings

- Linear models: interpretable but may underfit.

- KNN: simple, intuitive, noise-sensitive.

- Naive Bayes: fast, assumes independence.

-SVM: robust for complex datasets.

- Decision Tree: interpretable, may overfit.

Future Improvements

- Hyperparameter tuning, ensembles (Random Forest, XGBoost), feature engineering. - Cross-validation and model comparison charts placeholders.

End of Report