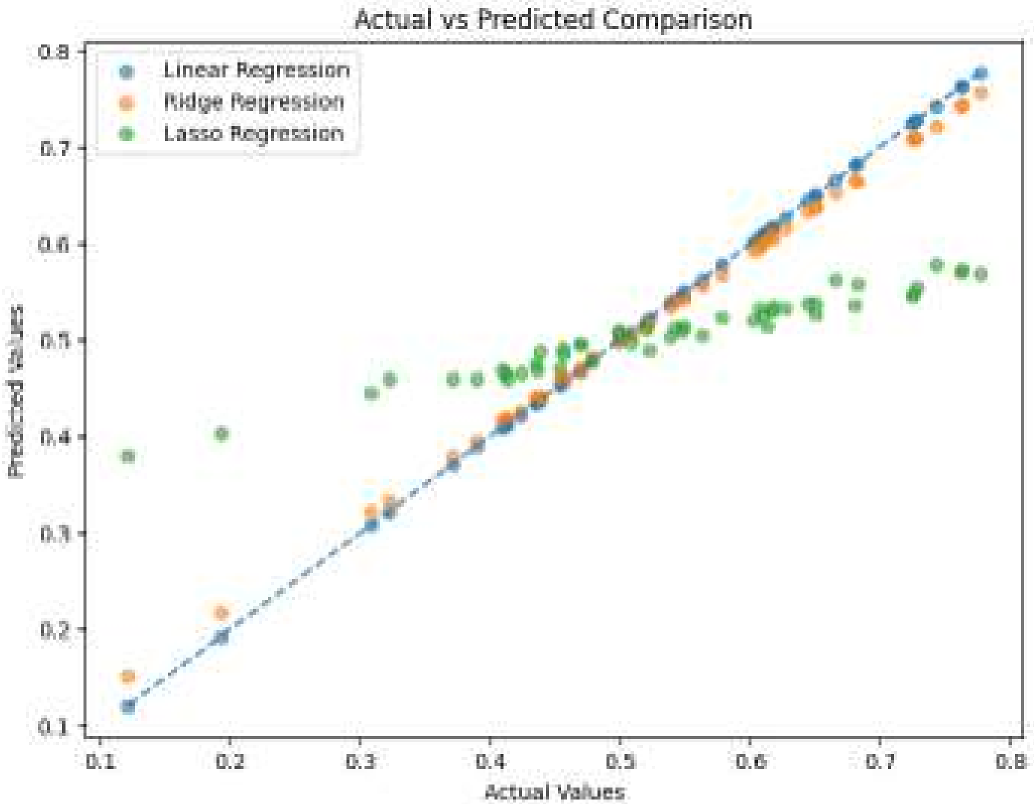
```python
plt.figure(figsize=(8,6))

plt.scatter(y_test, y_pred_lr, label="Linear Regression", alpha=0.6)
plt.scatter(y_test, y_pred_ridge, label="Ridge Regression", alpha=0.6)
plt.scatter(y_test, y_pred_lasso, label="Lasso Regression", alpha=0.6)

plt.plot([y_test.min(), y_test.max()],
         [y_test.min(), y_test.max()],
         linestyle='--')

plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.title("Actual vs Predicted Comparison")
plt.legend()
plt.show()
```

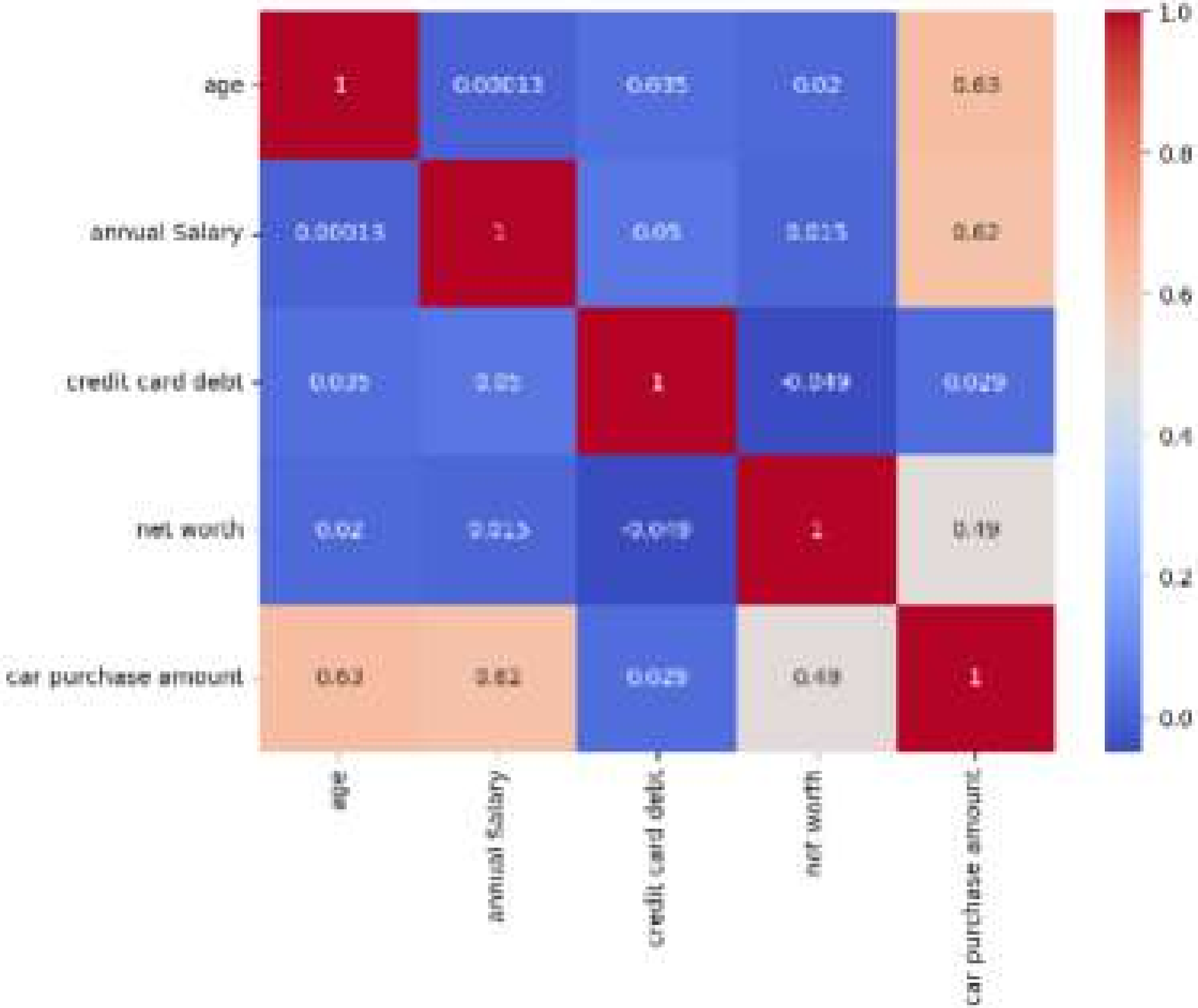


The plot shows that Linear and Ridge Regression predictions closely match the actual values, indicating high accuracy, while Lass Regression predictions deviate more, reflecting lower performance due to feature shrinkage.

Age and annual salary show strong positive correlation with car purchase amount

Credit card debt has very low correlation, so it has minimal impact

```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(8,6))
sns.heatmap(df2.corr(),annot=True,cmap="coolwarm")
plt.show()
```



```python
# Import required libraries
import pandas as pd
from textblob import TextBlob

# Step 1: Ensure review_content is string (IMPORTANT)
df['review_content'] = df['review_content'].astype(str)

# Step 2: Apply sentiment analysis using TextBlob
df['sentiment'] = df['review_content'].apply(
    lambda text: TextBlob(text).sentiment.polarity
)

# Step 3: Sort reviews by sentiment score
# Most positive reviews
positive_reviews = df.sort_values(by='sentiment', ascending=False)

# Most negative reviews
negative_reviews = df.sort_values(by='sentiment', ascending=True)

# Step 4: Display top 10 positive and negative reviews
top_positive = positive_reviews[['product_id', 'user_id', 'review_content', 'sentiment']].head(10)
top_negative = negative_reviews[['product_id', 'user_id', 'review_content', 'sentiment']].head(10)

print("Top 10 Positive Reviews:")
print(top_positive)

print("\nTop 10 Negative Reviews:")
print(top_negative)
```

```
Top 10 Positive Reviews:
      product_id  user_id review_content  sentiment
1464         134      433            475        0.0
0            346      623            604        0.0
1            848       88            413        0.0
2            819      849            674        0.0
3            643      254            269        0.0
4            588       17            120        0.0
5            771      210            518        0.0
6            761      662            123        0.0
7            614     1162           1122        0.0
1448         952       24           1006        0.0

Top 10 Negative Reviews:
      product_id  user_id review_content  sentiment
1455          53       82              5        0.0
1454         786      134            702        0.0
1453         348     1147           1062        0.0
1452        1245      755            644        0.0
1451         150      358             98        0.0
1450        1314      717            800        0.0
1449        1273      344            848        0.0
1448         952       24           1006        0.0
1447         258       62            144        0.0
1446         984      808            554        0.0
```

Answer 8: Discounted price and rating have a weak positive correlation. This means that products with higher discounted pri
have slightly higher ratings, but the relationship is not very strong.

```python
# Linear Regression
lr = LinearRegression()
lr.fit(x_train, y_train)
y_pred_lr = lr.predict(x_test)

# Ridge Regression
ridge = Ridge(alpha=1.0)
ridge.fit(x_train, y_train)
y_pred_ridge = ridge.predict(x_test)

# Lasso Regression
lasso = Lasso(alpha=0.01)
lasso.fit(x_train, y_train)
y_pred_lasso = lasso.predict(x_test)
```

3. Evaluation Metrics Function

```python
def evaluate_model(name, y_true, y_pred):
    mae = mean_absolute_error(y_true, y_pred)
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_true, y_pred)

    return [name, mae, mse, rmse, r2]
```

4. Evaluate All Models

```python
results = []

results.append(evaluate_model("Linear Regression", y_test, y_pred_lr))
results.append(evaluate_model("Ridge Regression", y_test, y_pred_ridge))
results.append(evaluate_model("Lasso Regression", y_test, y_pred_lasso))

results_df = pd.DataFrame(
    results,
    columns=["Model", "MAE", "MSE", "RMSE", "R2 Score"]
)

results_df
```

	Model	MAE	MSE	RMSE	R2 Score
0	Linear Regression	0.000016	3.992213e-10	0.000020	1.000000
1	Ridge Regression	0.009647	1.403898e-04	0.011849	0.993089
2	Lasso Regression	0.085623	1.135407e-02	0.106555	0.441032

```
       [0.54146119],
       [0.38927052],
       [0.62661214],
       [0.40908055],
       [0.37072477],
       [0.46885649],
       [0.60642838],
       [0.53860366]])
```

Train the model

```python
from sklearn.linear_model import LinearRegression
```

```python
model = LinearRegression()
model.fit(x_train, y_train)
```

```
LinearRegression
LinearRegression()
```

Make predictions

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

Evaluate the model

```python
from sklearn.metrics import mean_squared_error, r2_score
```

```python
print("MSE:", mean_squared_error(y_test, y_pred))
print("R2 Score:", r2_score(y_test, y_pred))
```

```
MSE: 3.9922134535627505e-10
R2 Score: 0.9999999883460608
```

MSE (Mean Squared Error) It measures how far the predicted values (y_pred) are from the actual values (y_test).

Your value: 3.99e-10 → extremely small, almost 0.

Interpretation: The predictions are almost perfect, almost exactly equal to the actual values.

R2 Score

Measures how well the model explains the variance in the target.

Range: 0 to 1 (sometimes negative if very bad).

Your value: 0.99999998 → almost 1, meaning the model explains nearly 100% of the variance.