In [5]:
```
import pandas as pd
data = pd.read_csv('C:\\Users\\123\\TCS\\german_credit_data.csv')

def classify_credit_risk(row):
    if row['Credit amount'] > 5000 and row['Saving accounts'] in ['little', 'moderate'
        return 1  # Good credit risk

    elif row['Credit amount'] <= 5000 and row['Saving accounts'] == 'no known savings'
        return 2  # Bad credit risk

    else:
        return 2

data['target'] = data.apply(classify_credit_risk, axis=1)

print(data[['Credit amount', 'Saving accounts', 'target']].head())
```

```
   Credit amount Saving accounts  target
0           1169             NaN       2
1           5951          little       1
2           2096          little       2
3           7882          little       1
4           4870          little       2
```

In [6]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler

data = pd.read_csv('C:\\Users\\123\\TCS\\german_credit_data.csv')


data['Saving accounts'] = data['Saving accounts'].fillna(data['Saving accounts'].mode(


data['target'] = data['Credit amount'].apply(lambda x: 1 if x > 5000 else 2)


categorical_columns = ['Sex', 'Job', 'Housing', 'Saving accounts', 'Checking account',
label_encoder = LabelEncoder()
for column in categorical_columns:
    data[column] = label_encoder.fit_transform(data[column])


scaler = StandardScaler()
data[['Credit amount', 'Duration']] = scaler.fit_transform(data[['Credit amount', 'Dur

X = data.drop(columns=['target'])
y = data['target']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=

print(X_train.head())
print(y_train.head())
```

```
     Unnamed: 0  Age  Sex  Job  Housing  Saving accounts  Checking account  \
29           29   63    1    2        1                0                 0
535         535   33    1    2        2                0                 2
695         695   50    1    2        2                2                 3
557         557   29    0    2        1                0                 3
836         836   21    0    2        1                0                 3

     Credit amount  Duration  Purpose
29        1.263499  3.243815        0
535      -0.337522  0.008048        3
695      -0.721384 -1.236478        1
557       0.613804  0.008048        1
836      -0.845439 -0.738668        5
29     1
535    2
695    2
557    1
836    2
Name: target, dtype: int64
```

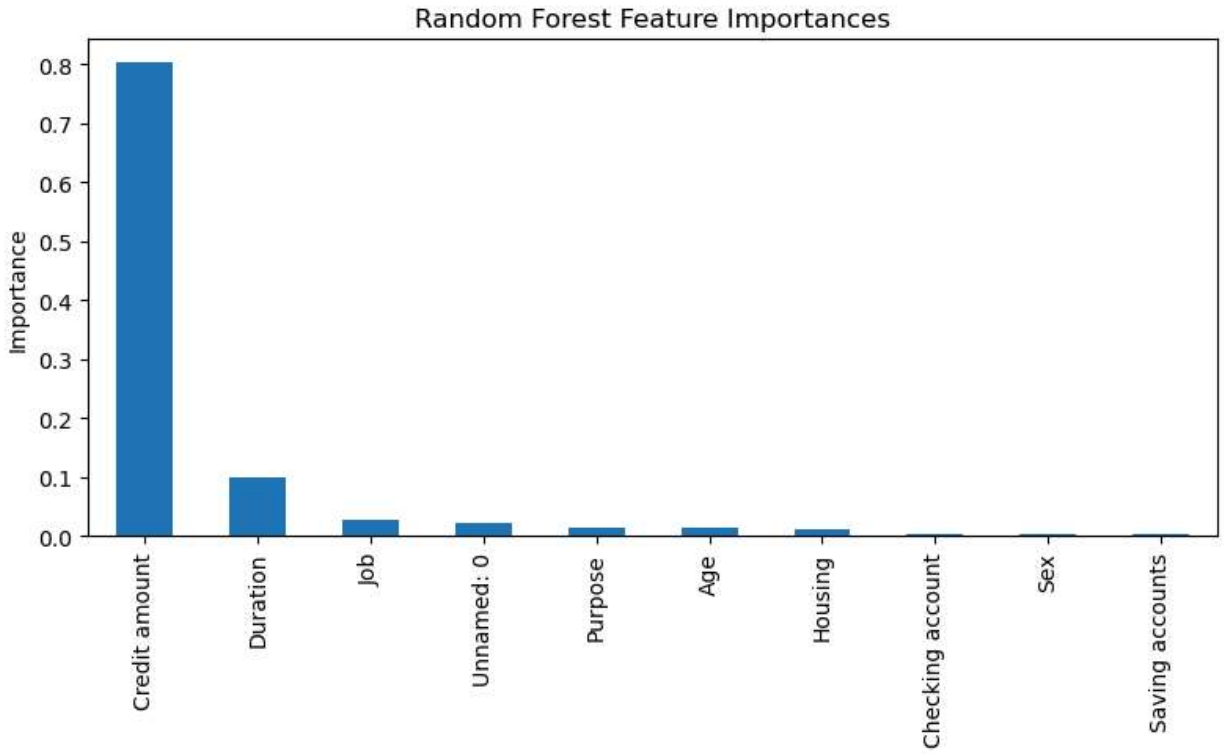
In [7]:

```python
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score, precision_score, re
from sklearn.model_selection import cross_val_score, StratifiedKFold

log_model = LogisticRegression(max_iter=1000)
rf_model = RandomForestClassifier(random_state=42, n_estimators=100)

log_model.fit(X_train, y_train)
rf_model.fit(X_train, y_train)

log_preds = log_model.predict(X_test)
rf_preds = rf_model.predict(X_test)

def evaluate_model(name, y_true, y_pred):
    print(f"\n{name} Evaluation:")
    print("Accuracy :", accuracy_score(y_true, y_pred))
    print("Precision:", precision_score(y_true, y_pred, pos_label=1))
    print("Recall   :", recall_score(y_true, y_pred, pos_label=1))
    print("F1 Score :", f1_score(y_true, y_pred, pos_label=1))
    print("\nClassification Report:\n", classification_report(y_true, y_pred))

evaluate_model("Logistic Regression", y_test, log_preds)
evaluate_model("Random Forest", y_test, rf_preds)

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
rf_cv_scores = cross_val_score(rf_model, X, y, cv=cv, scoring='f1_macro')
print("\nRandom Forest CV F1 Macro Score:", rf_cv_scores.mean())
```

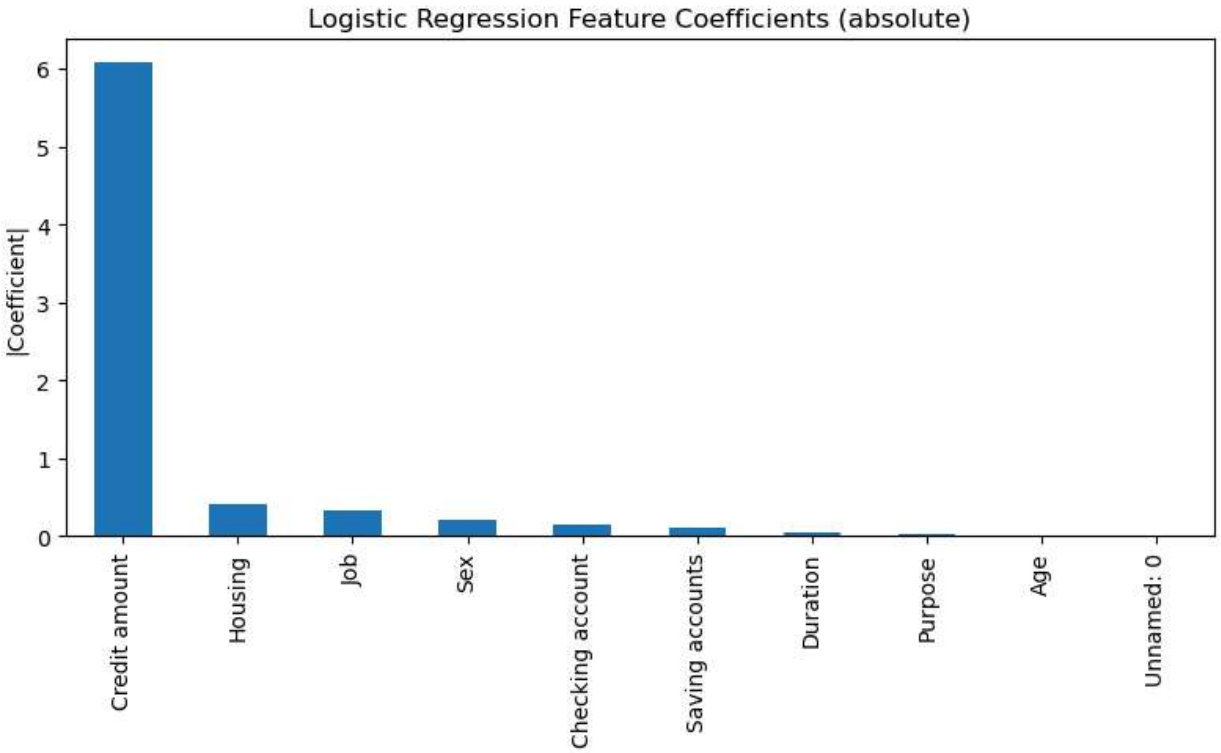
```
Logistic Regression Evaluation:
Accuracy : 1.0
Precision: 1.0
Recall   : 1.0
F1 Score : 1.0

Classification Report:
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        30
           2       1.00      1.00      1.00       170

    accuracy                           1.00       200
   macro avg       1.00      1.00      1.00       200
weighted avg       1.00      1.00      1.00       200


Random Forest Evaluation:
Accuracy : 1.0
Precision: 1.0
Recall   : 1.0
F1 Score : 1.0

Classification Report:
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        30
           2       1.00      1.00      1.00       170

    accuracy                           1.00       200
   macro avg       1.00      1.00      1.00       200
weighted avg       1.00      1.00      1.00       200


Random Forest CV F1 Macro Score: 1.0
```

In [13]:

```python
import matplotlib.pyplot as plt

importances = rf_model.feature_importances_
feat_imp = pd.Series(importances, index=X.columns).sort_values(ascending=False)

plt.figure(figsize=(8,5))
feat_imp.plot.bar()
plt.title("Random Forest Feature Importances")
plt.ylabel("Importance")
plt.tight_layout()
plt.show()
```

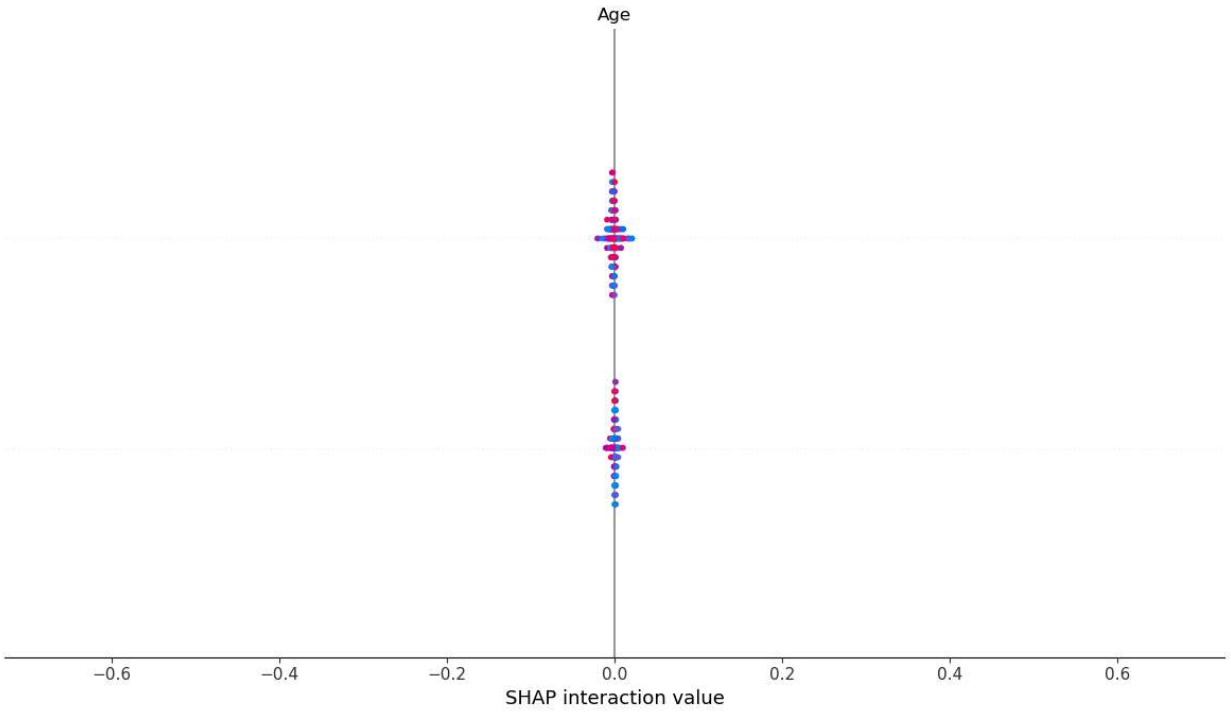


In [9]:
```
import matplotlib.pyplot as plt
import pandas as pd

coeffs = pd.Series(log_model.coef_[0], index=X.columns).abs().sort_values(ascending=Fa

plt.figure(figsize=(8,5))
coeffs.plot.bar()
plt.title("Logistic Regression Feature Coefficients (absolute)")
plt.ylabel("|Coefficient|")
plt.tight_layout()
plt.show()
```



In [10]:
```
import shap

explainer = shap.Explainer(rf_model, X_train)
shap_values = explainer(X_test)

shap.summary_plot(shap_values, X_test)
```

```
<Figure size 640x480 with 0 Axes>
```



In [ ]: