BLOOD DONATION FORECAST

1. Inspecting transfusion.data file

```
# inspect_data.py
def inspect_file(file_path):
    with open(file_path, 'r') as file:
        content = file.readlines()
        for line in content[:5]:  # Display the first 5 lines for
inspection
            print(line.strip())

if __name__ == "__main__":
    inspect_file('transfusion.data')
```

2. Loading the blood donations data

```
# load_data.py
import pandas as pd

def load_data(file_path):
    return pd.read_csv(file_path)

if __name__ == "__main__":
    df = load_data('transfusion.data')
    print(df.head())
```

3. Inspecting transfusion DataFrame

```
# load_data.py (continued)
def inspect_data(df):
    print(df.info())
    print(df.head())

if __name__ == "__main__":
    df = load_data('transfusion.data')
    inspect_data(df)
```

4. Creating target column

```
import pandas as pd

def create_target_column(df):
    df.columns = ['Recency', 'Frequency', 'Monetary', 'Time',
'Donated_Mar_2007']
    df['Target'] = df['Donated_Mar_2007']
    df.drop(columns='Donated_Mar_2007', inplace=True)
    return df

def check_target_incidence(df):
    print(df['Target'].value_counts(normalize=True))

if __name__ == "__main__":
    df = pd.read_csv('transfusion.data')
    df = create_target_column(df)
    check_target_incidence(df)
```

5. Checking target incidence

```
import pandas as pd

def create_target_column(df):
    df.columns = ['Recency', 'Frequency', 'Monetary', 'Time',
'Donated_Mar_2007']
    df['Target'] = df['Donated_Mar_2007']
    df.drop(columns='Donated_Mar_2007', inplace=True)
    return df

def check_target_incidence(df):
    print(df['Target'].value_counts(normalize=True))

if __name__ == "__main__":
    df = pd.read_csv('transfusion.data')
    df = create_target_column(df)
    check_target_incidence(df)
```

6. Splitting transfusion into train and test datasets

```
import pandas as pd
from sklearn.model_selection import train_test_split

def split_data(df):
    X = df.drop(columns='Target')
    y = df['Target']
    return train_test_split(X, y, test_size=0.25, random_state=42,
stratify=y)

if __name__ == "__main__":
    df = pd.read_csv('transfusion.data')
    from preprocess_data import create_target_column  # Ensure correct
import
    df = create_target_column(df)
    X_train, X_test, y_train, y_test = split_data(df)
    print(f"Train size: {X_train.shape}, Test size: {X_test.shape}")
```

7. Selecting model using TPOT

```
import pandas as pd
from preprocess_data import create_target_column
from split_data import split_data
from tpot import TPOTClassifier

def train_tpot(X_train, y_train):
    tpot = TPOTClassifier(verbosity=2, generations=5, population_size=20,
random_state=42, config_dict='TPOT sparse')
    tpot.fit(X_train, y_train)
    return tpot

if __name__ == "__main__":
    df = pd.read_csv('transfusion.data')
    df = create_target_column(df)
    X_train, X_test, y_train, y_test = split_data(df)
    tpot_model = train_tpot(X_train, y_train)
```

8. Checking the variance

```
import pandas as pd
from preprocess_data import create_target_column
from split_data import split_data

def check_variance(X_train):
    print(X_train.var())

if __name__ == "__main__":
    df = pd.read_csv('transfusion.data')
    df = create_target_column(df)
    X_train, X_test, y_train, y_test = split_data(df)
    check_variance(X_train)
```

9. Log normalization

```
import numpy as np
import pandas as pd
from preprocess_data import create_target_column
from split_data import split_data

def log_normalize(X_train, X_test):
    X_train_log = X_train.copy()
    X_test_log = X_test.copy()
    for column in X_train_log.columns:
        X_train_log[column] = np.log1p(X_train_log[column])
        X_test_log[column] = np.log1p(X_test_log[column])
    return X_train_log, X_test_log

if __name__ == "__main__":
    df = pd.read_csv('transfusion.data')
    df = create_target_column(df)
    X_train, X_test, y_train, y_test = split_data(df)
    X_train_log, X_test_log = log_normalize(X_train, X_test)
    print(X_train_log.head())
```

10. Training the linear regression model

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix

def train_logistic_regression(X_train_log, y_train, X_test_log, y_test):
    log_reg = LogisticRegression(max_iter=1000)
    log_reg.fit(X_train_log, y_train)
    y_pred = log_reg.predict(X_test_log)
    accuracy = accuracy_score(y_test, y_pred)
    conf_matrix = confusion_matrix(y_test, y_pred)
    return accuracy, conf_matrix

if __name__ == "__main__":
    import pandas as pd
    from preprocess_data import create_target_column
    from split_data import split_data
    from log_normalize import log_normalize

    df = pd.read_csv('transfusion.data')
    df = create_target_column(df)
    X_train, X_test, y_train, y_test = split_data(df)
    X_train_log, X_test_log = log_normalize(X_train, X_test)
    accuracy, conf_matrix = train_logistic_regression(X_train_log, y_train, 
X_test_log, y_test)
    print(f'Accuracy: {accuracy}')
    print(f'Confusion Matrix:\n{conf_matrix}')
```

11. Conclusion

```
import pandas as pd
from split_data import split_data
from preprocess_data import create_target_column
from log_normalize import log_normalize
from train_tpot import train_tpot
from train_logistic_regression import train_logistic_regression

def load_data(file_path):
    return pd.read_csv(file_path)

def main():
    df = load_data('transfusion.data')
    df = create_target_column(df)
    X_train, X_test, y_train, y_test = split_data(df)
    X_train_log, X_test_log = log_normalize(X_train, X_test)
    tpot_model = train_tpot(X_train, y_train)
    accuracy, conf_matrix = train_logistic_regression(X_train_log, y_train, 
X_test_log, y_test)
    print(f'Accuracy: {accuracy}')
    print(f'Confusion Matrix:\n{conf_matrix}')

if __name__ == "__main__":
    main()
```

This project guides you through the entire process of building a predictive model for blood donations, starting from