```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import statsmodels.api as sm
```

```
data = pd.read_excel('creditscore.xlsx')
```

```
data.head()
```

	BureauInquiries	CreditUsage	TotalCredit	CollectionReports	MissedPayments	HomeOwr
0	7	0.27	18000	0	2	
1	7	0.23	16000	0	1	
2	7	0.27	18000	0	1	
3	8	0.23	21000	0	0	
4	8	0.42	32000	0	2	

```
data.isnull().sum()
```

```
BureauInquiries      0
CreditUsage          0
TotalCredit          0
CollectionReports    0
MissedPayments       0
HomeOwner            0
CreditAge            0
TimeOnJob            0
Repay                0
Partition            0
dtype: int64
```

```
data.duplicated().sum()
```

```
214
```

```
data.drop_duplicates(inplace=True)
```

```
data = data.drop('Partition', axis=1)
X = data.drop('Repay', axis=1)
y = data['Repay']
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)
```

FULL MODEL- All predictors used

```
full_model = LogisticRegression()
full_model.fit(X_train, y_train)

y_pred = full_model.predict(X_test)
full_model_accuracy = accuracy_score(y_test, y_pred)
print(full_model_accuracy)
```

```
0.7088948787061995
```

REDUCED MODEL- Only 4 predictors used

```
reduced_model = LogisticRegression()
reduced_model.fit(X_train[['CreditUsage', 'MissedPayments', 'HomeOwner', 'CreditAge']], y_tra

y_pred = reduced_model.predict(X_test[['CreditUsage', 'MissedPayments', 'HomeOwner', 'CreditA
reduced_model_accuracy = accuracy_score(y_test, y_pred)
print(reduced_model_accuracy)
```

```
0.7196765498652291
```

SIGNIFICANT MODEL- Significant predictors used (using statsmodel library)

```
# We fit the logistic regression model twice because we first need to identify the significan

X = sm.add_constant(X) # const term added to incl intercept term (intercept term is value of
model = sm.Logit(y, X).fit(disp=False) # fit logistic regression
significant_predictors = list(model.pvalues[model.pvalues < 0.05].index) # get significant pr
significant_model = sm.Logit(y, X[significant_predictors]).fit(disp=False) # fit significant

y_pred = significant_model.predict(X_test[significant_predictors])
y_pred_class = [1 if p > 0.5 else 0 for p in y_pred]
signficant_model_accuracy = accuracy_score(y_test, y_pred_class)
print(signficant_model_accuracy)
```

```
0.7358490566037735
```

Accuracy of all three models

```
df = pd.DataFrame({'Full model': [full_model_accuracy], 'Reduced Model': [reduced_model_accur
df
```

	Full model	Reduced Model	Signficant Model
0	0.708895	0.719677	0.735849

Expressing significant model as a mathematical equation

```
#extract coeff
coeff = significant_model.params.apply(lambda x: "{:.5f}".format(x))

# construct eqn
equation = 'p = 1 / (1 + exp(-(' + str(coeff[0]) + ' + '

for i in range(1, len(coeff)):
    equation += str(coeff[i]) + ' * X' + str(i) + ' + '

# remove last '+' and add 3 closing paranthesis
equation = equation[:-3] + ')))'

print(equation)
```

```
p = 1 / (1 + exp(-(-0.09569 + -3.67115 * X1 + -1.62960 * X2 + -0.56513 * X3 + 0.12729 *
```