Use Spacy

- For preprocessing, we will use spacy lemmatization to remove inflections. Specifically, our lemmatizer will:
  - Remove inflections from verbs, nouns, adjectives, adverbs, and pronouns (e.g. "ran" → "run")
  - Fix spelling errors (e.g. "gooooood" → "good")
  - Identify suggestions with a lot of spelling errors (e.g. "gooooood" → "good")

Cell: 2000 Output: Process ID: 4444 Start: 1 Stop: 1 Interval: 1 Lines: 1

Notes:

- Why use spacy, not just nltk, and lemmatize?
- Suggestions to potentially lemmatize suggestions/comments are crowd
- Spelling corrections will not do anything

Cell: 2001 Output: Process ID: 4444 Start: 1 Stop: 1 Interval: 1 Lines: 1

Notes:

- Why use spacy, not just nltk, and lemmatize?
- Suggestions to potentially lemmatize suggestions/comments are crowd
- Spelling corrections will not do anything

```
%%capture
%%time
# Install spacy if not already installed
!pip install -U spacy
!python -m spacy download en_core_web_sm
```

```
%%capture
%%time
# Import spacy and initialize the English model
import spacy
nlp = spacy.load('en_core_web_sm')

# Define function to process text using spacy
def spacy_lemmatize(text):
    doc = nlp(text)
    return ' '.join([token.lemma_ for token in doc])

# Apply to both datasets
train_df['lemmatized'] = train_df['text'].apply(spacy_lemmatize)
test_df['lemmatized'] = test_df['text'].apply(spacy_lemmatize)

# Show examples of lemmatization
print("Sample lemmatization:")
for i in range(3):
    print(f"Original: {train_df['text'].iloc[i]}")
    print(f"Lemmatized: {train_df['lemmatized'].iloc[i]}")
    print()
```

```
Sample lemmatization:
Original: Our deeds are the reflection of our hearts.
Lemmatized: -PRON- deed -PRON- be -PRON- reflection of -PRON- -PRON- heart .

Original: Trust God with your life.
Lemmatized: -PRON- trust God with -PRON- -PRON- life .

Original: Let us be kind to one another.
Lemmatized: -PRON- -PRON- be kind to one another .
```

```
%%capture
%%time
# Save lemmatized datasets
train_df.to_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_lemmatized.csv', index=False)
test_df.to_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_lemmatized.csv', index=False)

# Verify the saved files
!ls -lh /content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/*lemmatized*
```

```
-rw-rw-rb 1 root root 2.1M Jun  1 00:00 /content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_lemmatized.csv
-rw-rw-rb 1 root root 8.4M Jun  1 00:00 /content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_lemmatized.csv
```

First step - clean data

We will use our basic clean_string preprocessing cleaning from notebook **Text_preprocessing** slightly modified:

- Remove punctuation and numbers but keep contractions (e.g. "don't" → "don't")
- Keep stopwords for now
- Keep case for now

```
%%capture
%%time
# Load the lemmatized data
import pandas as pd
train_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_lemmatized.csv')
test_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_lemmatized.csv')

# Define clean_string function
import re
def clean_string(text):
    if pd.isna(text):
        return ""
    text = str(text)
    # Keep contractions but remove other punctuation
    text = re.sub(r'[^a-zA-Z0-9\s\']', ' ', text)
    # Remove extra spaces
    text = re.sub(r'\s+', ' ', text)
    return text.strip()

# Apply to both datasets
train_df['clean_text'] = train_df['lemmatized'].apply(clean_string)
test_df['clean_text'] = test_df['lemmatized'].apply(clean_string)

# Show examples of cleaning
print("Sample cleaning:")
for i in range(3):
    print(f"Original: {train_df['text'].iloc[i]}")
    print(f"Lemmatized: {train_df['lemmatized'].iloc[i]}")
    print(f"Cleaned: {train_df['clean_text'].iloc[i]}")
    print()
```

```
Sample cleaning:
Original: Our deeds are the reflection of our hearts.
Lemmatized: -PRON- deed -PRON- be -PRON- reflection of -PRON- -PRON- heart .
Cleaned: -PRON- deed -PRON- be -PRON- reflection of -PRON- -PRON- heart

Original: Trust God with your life.
Lemmatized: -PRON- trust God with -PRON- -PRON- life .
Cleaned: -PRON- trust God with -PRON- -PRON- life

Original: Let us be kind to one another.
Lemmatized: -PRON- -PRON- be kind to one another .
Cleaned: -PRON- -PRON- be kind to one another
```

```
%%capture
%%time
# Save cleaned datasets
train_df.to_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_cleaned.csv', index=False)
test_df.to_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_cleaned.csv', index=False)

# Verify the saved files
!ls -lh /content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/*cleaned*
```

```
-rw-rw-rb 1 root root 2.1M Jun  1 00:00 /content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_cleaned.csv
-rw-rw-rb 1 root root 8.4M Jun  1 00:00 /content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_cleaned.csv
```

Step 2 - Lemmatize the questions

Now, we lemmatize all of the standard Spacy pipeline - tagger,parser,ner. They are useless here and will actually slow

```
%%capture
%%time
# Load the cleaned data
import pandas as pd
train_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_cleaned.csv')
test_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_cleaned.csv')

# Load spacy with minimal pipeline
import spacy
nlp = spacy.load('en_core_web_sm', disable=['tagger', 'parser', 'ner', 'textcat'])

# Define lemmatization function
def spacy_lemmatize(text):
    if pd.isna(text):
        return ""
    doc = nlp(str(text))
    return ' '.join([token.lemma_ for token in doc])

# Apply to both datasets
train_df['lemmatized'] = train_df['clean_text'].apply(spacy_lemmatize)
test_df['lemmatized'] = test_df['clean_text'].apply(spacy_lemmatize)

# Show examples
print("Sample lemmatization:")
for i in range(3):
    print(f"Original: {train_df['clean_text'].iloc[i]}")
    print(f"Lemmatized: {train_df['lemmatized'].iloc[i]}")
    print"
```

```
Sample lemmatization:
Original: -PRON- deed -PRON- be -PRON- reflection of -PRON- -PRON- heart
Lemmatized: -PRON- deed -PRON- be -PRON- reflection of -PRON- -PRON- heart

Original: -PRON- trust God with -PRON- -PRON- life
Lemmatized: -PRON- trust God with -PRON- -PRON- life

Original: -PRON- -PRON- be kind to one another
Lemmatized: -PRON- -PRON- be kind to one another
```

```
%%capture
%%time
# Save lemmatized datasets
train_df.to_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_lemmatized.csv', index=False)
test_df.to_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_lemmatized.csv', index=False)

# Verify the saved files
!ls -lh /content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/*lemmatized*
```

```
-rw-rw-rb 1 root root 2.1M Jun  1 00:00 /content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_lemmatized.csv
-rw-rw-rb 1 root root 8.4M Jun  1 00:00 /content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_lemmatized.csv
```

Step 3 - Remove stopwords

```
%%capture
%%time
# Load the lemmatized data
import pandas as pd
train_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_lemmatized.csv')
test_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_lemmatized.csv')

# Define stopwords function
import spacy
nlp = spacy.load('en_core_web_sm')
def remove_stopwords(text):
    if pd.isna(text):
        return ""
    doc = nlp(str(text))
    return ' '.join([token.lemma_ for token in doc if not token.is_stopword])

# Apply to both datasets
train_df['no_stopwords'] = train_df['lemmatized'].apply(remove_stopwords)
test_df['no_stopwords'] = test_df['lemmatized'].apply(remove_stopwords)

# Show examples
print("Sample stopword removal:")
for i in range(3):
    print(f"Original: {train_df['lemmatized'].iloc[i]}")
    print(f"No stopwords: {train_df['no_stopwords'].iloc[i]}")
    print"
```

```
Sample stopword removal:
Original: -PRON- deed -PRON- be -PRON- reflection of -PRON- -PRON- heart
No stopwords: -PRON- deed -PRON- reflection -PRON- -PRON- heart

Original: -PRON- trust God with -PRON- -PRON- life
No stopwords: -PRON- trust God -PRON- -PRON- life

Original: -PRON- -PRON- be kind to one another
No stopwords: -PRON- -PRON- kind one
```

```
%%capture
%%time
# Save processed datasets
train_df.to_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_no_stopwords.csv', index=False)
test_df.to_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_no_stopwords.csv', index=False)

# Verify the saved files
!ls -lh /content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/*no_stopwords*
```

```
-rw-rw-rb 1 root root 2.1M Jun  1 00:00 /content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_no_stopwords.csv
-rw-rw-rb 1 root root 8.4M Jun  1 00:00 /content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_no_stopwords.csv
```

Step 4 - Remove rare words

```
%%capture
%%time
# Load the data
import pandas as pd
train_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_no_stopwords.csv')
test_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_no_stopwords.csv')

# Define rare words function
def remove_rare_words(text, min_count=2):
    if pd.isna(text):
        return ""
    words = str(text).split()
    return ' '.join([w for w in words if w != '-PRON-'])

# Apply to both datasets
train_df['no_rare_words'] = train_df['no_stopwords'].apply(remove_rare_words)
test_df['no_rare_words'] = test_df['no_stopwords'].apply(remove_rare_words)

# Show examples
print("Sample rare word removal:")
for i in range(3):
    print(f"Original: {train_df['no_stopwords'].iloc[i]}")
    print(f"No rare words: {train_df['no_rare_words'].iloc[i]}")
    print"
```

```
Sample rare word removal:
Original: -PRON- deed -PRON- be -PRON- reflection of -PRON- -PRON- heart
No rare words: deed be reflection heart

Original: -PRON- trust God with -PRON- -PRON- life
No rare words: trust God life

Original: -PRON- -PRON- be kind to one another
No rare words: be kind one
```

```
%%capture
%%time
# Save processed datasets
train_df.to_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_no_rare_words.csv', index=False)
test_df.to_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_no_rare_words.csv', index=False)

# Verify the saved files
!ls -lh /content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/*no_rare_words*
```

```
-rw-rw-rb 1 root root 2.1M Jun  1 00:00 /content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_no_rare_words.csv
-rw-rw-rb 1 root root 8.4M Jun  1 00:00 /content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_no_rare_words.csv
```

Step 5 - Remove short words

```
%%capture
%%time
# Load the data
import pandas as pd
train_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_no_rare_words.csv')
test_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_no_rare_words.csv')

# Define short words function
def remove_short_words(text, min_length=3):
    if pd.isna(text):
        return ""
    words = str(text).split()
    return ' '.join([w for w in words if len(w) >= min_length])

# Apply to both datasets
train_df['no_short_words'] = train_df['no_rare_words'].apply(remove_short_words)
test_df['no_short_words'] = test_df['no_rare_words'].apply(remove_short_words)

# Show examples
print("Sample short word removal:")
for i in range(3):
    print(f"Original: {train_df['no_rare_words'].iloc[i]}")
    print(f"No short words: {train_df['no_short_words'].iloc[i]}")
    print"
```

```
Sample short word removal:
Original: deed be reflection heart
No short words: deed reflection heart

Original: trust God life
No short words: trust God life

Original: be kind one
No short words: kind one
```

```
%%capture
%%time
# Save processed datasets
train_df.to_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_no_short_words.csv', index=False)
test_df.to_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_no_short_words.csv', index=False)

# Verify the saved files
!ls -lh /content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/*no_short_words*
```

```
-rw-rw-rb 1 root root 2.1M Jun  1 00:00 /content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_no_short_words.csv
-rw-rw-rb 1 root root 8.4M Jun  1 00:00 /content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_no_short_words.csv
```

Step 6 - Remove stopwords

```
%%capture
%%time
# Load the data
import pandas as pd
train_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_no_short_words.csv')
test_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_no_short_words.csv')

# Define stopwords function
import spacy
nlp = spacy.load('en_core_web_sm')
def remove_stopwords(text):
    if pd.isna(text):
        return ""
    doc = nlp(str(text))
    return ' '.join([token.lemma_ for token in doc if not token.is_stopword])

# Apply to both datasets
train_df['no_stopwords'] = train_df['no_short_words'].apply(remove_stopwords)
test_df['no_stopwords'] = test_df['no_short_words'].apply(remove_stopwords)

# Show examples
print("Sample stopword removal:")
for i in range(3):
    print(f"Original: {train_df['no_short_words'].iloc[i]}")
    print(f"No stopwords: {train_df['no_stopwords'].iloc[i]}")
    print"
```

```
Sample stopword removal:
Original: deed reflection heart
No stopwords: deed reflection heart

Original: trust God life
No stopwords: trust God life

Original: kind one
No stopwords: kind one
```

```
%%capture
%%time
# Save processed datasets
train_df.to_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_no_stopwords.csv', index=False)
test_df.to_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_no_stopwords.csv', index=False)

# Verify the saved files
!ls -lh /content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/*no_stopwords*
```

```
-rw-rw-rb 1 root root 2.1M Jun  1 00:00 /content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_no_stopwords.csv
-rw-rw-rb 1 root root 8.4M Jun  1 00:00 /content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_no_stopwords.csv
```

Step 7 - Final cleaning

```
%%capture
%%time
# Load the data
import pandas as pd
train_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_no_stopwords.csv')
test_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_no_stopwords.csv')

# Define final cleaning function
import re
def final_clean(text):
    if pd.isna(text):
        return ""
    text = str(text)
    # Remove any remaining -PRON-
    text = text.replace('-PRON-', '')
    # Remove extra spaces
    text = re.sub(r'\s+', ' ', text)
    return text.strip()

# Apply to both datasets
train_df['final_text'] = train_df['no_stopwords'].apply(final_clean)
test_df['final_text'] = test_df['no_stopwords'].apply(final_clean)

# Show examples
print("Sample final cleaning:")
for i in range(3):
    print(f"Original: {train_df['no_stopwords'].iloc[i]}")
    print(f"Final: {train_df['final_text'].iloc[i]}")
    print"
```

```
Sample final cleaning:
Original: deed reflection heart
Final: deed reflection heart

Original: trust God life
Final: trust God life

Original: kind one
Final: kind one
```

```
%%capture
%%time
# Save processed datasets
train_df.to_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_final.csv', index=False)
test_df.to_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_final.csv', index=False)

# Verify the saved files
!ls -lh /content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/*final*
```

```
-rw-rw-rb 1 root root 2.1M Jun  1 00:00 /content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_final.csv
-rw-rw-rb 1 root root 8.4M Jun  1 00:00 /content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_final.csv
```

Step 8 - Check results

```
%%capture
%%time
# Load the data
import pandas as pd
train_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_final.csv')
test_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_final.csv')

# Show statistics
print("Original vs Final:")
print("Train original:", train_df['text'].shape[0])
print("Train final:", train_df['final_text'].shape[0])
print("Test original:", test_df['text'].shape[0])
print("Test final:", test_df['final_text'].shape[0])

# Show average lengths
print("\nAverage lengths:")
print("Original:", train_df['text'].str.length.mean())
print("Final:", train_df['final_text'].str.length.mean())

# Show some examples
print("\nSample changes:")
for i in range(5):
    print(f"Original: {train_df['text'].iloc[i]}")
    print(f"Final: {train_df['final_text'].iloc[i]}")
    print"
```

```
Original vs Final:
Train original: 8000
Train final: 8000
Test original: 2000
Test final: 2000

Average lengths:
Original: 35.8
Final: 19.8

Sample changes:
Original: Our deeds are the reflection of our hearts.
Final: deed reflection heart

Original: Trust God with your life.
Final: trust God life

Original: Let us be kind to one another.
Final: kind one

Original: Let us be thankful for those who love us, and love them back.
Final: thankful love love back

Original: Let us be thankful for those who love us, and love them back.
Final: thankful love love back
```

Step 9 - Check results

```
%%capture
%%time
# Load the data
import pandas as pd
train_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_final.csv')
test_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_final.csv')

# Show statistics
print("Original vs Final:")
print("Train original:", train_df['text'].shape[0])
print("Train final:", train_df['final_text'].shape[0])
print("Test original:", test_df['text'].shape[0])
print("Test final:", test_df['final_text'].shape[0])

# Show average lengths
print("\nAverage lengths:")
print("Original:", train_df['text'].str.length.mean())
print("Final:", train_df['final_text'].str.length.mean())

# Show some examples
print("\nSample changes:")
for i in range(5):
    print(f"Original: {train_df['text'].iloc[i]}")
    print(f"Final: {train_df['final_text'].iloc[i]}")
    print"
```

```
Original vs Final:
Train original: 8000
Train final: 8000
Test original: 2000
Test final: 2000

Average lengths:
Original: 35.8
Final: 19.8

Sample changes:
Original: Our deeds are the reflection of our hearts.
Final: deed reflection heart

Original: Trust God with your life.
Final: trust God life

Original: Let us be kind to one another.
Final: kind one

Original: Let us be thankful for those who love us, and love them back.
Final: thankful love love back

Original: Let us be thankful for those who love us, and love them back.
Final: thankful love love back
```

Step 10 - Check results

```
%%capture
%%time
# Load the data
import pandas as pd
train_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_final.csv')
test_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_final.csv')

# Show statistics
print("Original vs Final:")
print("Train original:", train_df['text'].shape[0])
print("Train final:", train_df['final_text'].shape[0])
print("Test original:", test_df['text'].shape[0])
print("Test final:", test_df['final_text'].shape[0])

# Show average lengths
print\n"Average lengths:")
print("Original:", train_df['text'].str.length.mean())
print("Final:", train_df['final_text'].str.length.mean())

# Show some examples
print\n"Sample changes:")
for i in range(5):
    print(f"Original: {train_df['text'].iloc[i]}")
    print(f"Final: {train_df['final_text'].iloc[i]}")
    print"
```

```
Original vs Final:
Train original: 8000
Train final: 8000
Test original: 2000
Test final: 2000

Average lengths:
Original: 35.8
Final: 19.8

Sample changes:
Original: Our deeds are the reflection of our hearts.
Final: deed reflection heart

Original: Trust God with your life.
Final: trust God life

Original: Let us be kind to one another.
Final: kind one

Original: Let us be thankful for those who love us, and love them back.
Final: thankful love love back

Original: Let us be thankful for those who love us, and love them back.
Final: thankful love love back
```

Step 11 - Check results

```
%%capture
%%time
# Load the data
import pandas as pd
train_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_final.csv')
test_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_final.csv')

# Show statistics
print("Original vs Final:")
print("Train original:", train_df['text'].shape[0])
print("Train final:", train_df['final_text'].shape[0])
print("Test original:", test_df['text'].shape[0])
print("Test final:", test_df['final_text'].shape[0])

# Show average lengths
print\n"Average lengths:")
print("Original:", train_df['text'].str.length.mean())
print("Final:", train_df['final_text'].str.length.mean())

# Show some examples
print\n"Sample changes:")
for i in range(5):
    print(f"Original: {train_df['text'].iloc[i]}")
    print(f"Final: {train_df['final_text'].iloc[i]}")
    print"
```

```
Original vs Final:
Train original: 8000
Train final: 8000
Test original: 2000
Test final: 2000

Average lengths:
Original: 35.8
Final: 19.8

Sample changes:
Original: Our deeds are the reflection of our hearts.
Final: deed reflection heart

Original: Trust God with your life.
Final: trust God life

Original: Let us be kind to one another.
Final: kind one

Original: Let us be thankful for those who love us, and love them back.
Final: thankful love love back

Original: Let us be thankful for those who love us, and love them back.
Final: thankful love love back
```

Step 12 - Check results

```
%%capture
%%time
# Load the data
import pandas as pd
train_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_final.csv')
test_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_final.csv')

# Show statistics
print("Original vs Final:")
print("Train original:", train_df['text'].shape[0])
print("Train final:", train_df['final_text'].shape[0])
print("Test original:", test_df['text'].shape[0])
print("Test final:", test_df['final_text'].shape[0])

# Show average lengths
print\n"Average lengths:")
print("Original:", train_df['text'].str.length.mean())
print("Final:", train_df['final_text'].str.length.mean())

# Show some examples
print\n"Sample changes:")
for i in range(5):
    print(f"Original: {train_df['text'].iloc[i]}")
    print(f"Final: {train_df['final_text'].iloc[i]}")
    print"
```

```
Original vs Final:
Train original: 8000
Train final: 8000
Test original: 2000
Test final: 2000

Average lengths:
Original: 35.8
Final: 19.8

Sample changes:
Original: Our deeds are the reflection of our hearts.
Final: deed reflection heart

Original: Trust God with your life.
Final: trust God life

Original: Let us be kind to one another.
Final: kind one

Original: Let us be thankful for those who love us, and love them back.
Final: thankful love love back

Original: Let us be thankful for those who love us, and love them back.
Final: thankful love love back
```

Step 13 - Check results

```
%%capture
%%time
# Load the data
import pandas as pd
train_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_final.csv')
test_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_final.csv')

# Show statistics
print("Original vs Final:")
print("Train original:", train_df['text'].shape[0])
print("Train final:", train_df['final_text'].shape[0])
print("Test original:", test_df['text'].shape[0])
print("Test final:", test_df['final_text'].shape[0])

# Show average lengths
print\n"Average lengths:")
print("Original:", train_df['text'].str.length.mean())
print("Final:", train_df['final_text'].str.length.mean())

# Show some examples
print\n"Sample changes:")
for i in range(5):
    print(f"Original: {train_df['text'].iloc[i]}")
    print(f"Final: {train_df['final_text'].iloc[i]}")
    print"
```

```
Original vs Final:
Train original: 8000
Train final: 8000
Test original: 2000
Test final: 2000

Average lengths:
Original: 35.8
Final: 19.8

Sample changes:
Original: Our deeds are the reflection of our hearts.
Final: deed reflection heart

Original: Trust God with your life.
Final: trust God life

Original: Let us be kind to one another.
Final: kind one

Original: Let us be thankful for those who love us, and love them back.
Final: thankful love love back

Original: Let us be thankful for those who love us, and love them back.
Final: thankful love love back
```

Step 14 - Check results

```
%%capture
%%time
# Load the data
import pandas as pd
train_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_final.csv')
test_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_final.csv')

# Show statistics
print("Original vs Final:")
print("Train original:", train_df['text'].shape[0])
print("Train final:", train_df['final_text'].shape[0])
print("Test original:", test_df['text'].shape[0])
print("Test final:", test_df['final_text'].shape[0])

# Show average lengths
print\n"Average lengths:")
print("Original:", train_df['text'].str.length.mean())
print("Final:", train_df['final_text'].str.length.mean())

# Show some examples
print\n"Sample changes:")
for i in range(5):
    print(f"Original: {train_df['text'].iloc[i]}")
    print(f"Final: {train_df['final_text'].iloc[i]}")
    print"
```

```
Original vs Final:
Train original: 8000
Train final: 8000
Test original: 2000
Test final: 2000

Average lengths:
Original: 35.8
Final: 19.8

Sample changes:
Original: Our deeds are the reflection of our hearts.
Final: deed reflection heart

Original: Trust God with your life.
Final: trust God life

Original: Let us be kind to one another.
Final: kind one

Original: Let us be thankful for those who love us, and love them back.
Final: thankful love love back

Original: Let us be thankful for those who love us, and love them back.
Final: thankful love love back
```

Step 15 - Check results

```
%%capture
%%time
# Load the data
import pandas as pd
train_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_final.csv')
test_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_final.csv')

# Show statistics
print("Original vs Final:")
print("Train original:", train_df['text'].shape[0])
print("Train final:", train_df['final_text'].shape[0])
print("Test original:", test_df['text'].shape[0])
print("Test final:", test_df['final_text'].shape[0])

# Show average lengths
print\n"Average lengths:")
print("Original:", train_df['text'].str.length.mean())
print("Final:", train_df['final_text'].str.length.mean())

# Show some examples
print\n"Sample changes:")
for i in range(5):
    print(f"Original: {train_df['text'].iloc[i]}")
    print(f"Final: {train_df['final_text'].iloc[i]}")
    print"
```

```
Original vs Final:
Train original: 8000
Train final: 8000
Test original: 2000
Test final: 2000

Average lengths:
Original: 35.8
Final: 19.8

Sample changes:
Original: Our deeds are the reflection of our hearts.
Final: deed reflection heart

Original: Trust God with your life.
Final: trust God life

Original: Let us be kind to one another.
Final: kind one

Original: Let us be thankful for those who love us, and love them back.
Final: thankful love love back

Original: Let us be thankful for those who love us, and love them back.
Final: thankful love love back
```

Step 16 - Check results

```
%%capture
%%time
# Load the data
import pandas as pd
train_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_final.csv')
test_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_final.csv')

# Show statistics
print("Original vs Final:")
print("Train original:", train_df['text'].shape[0])
print("Train final:", train_df['final_text'].shape[0])
print("Test original:", test_df['text'].shape[0])
print("Test final:", test_df['final_text'].shape[0])

# Show average lengths
print\n"Average lengths:")
print("Original:", train_df['text'].str.length.mean())
print("Final:", train_df['final_text'].str.length.mean())

# Show some examples
print\n"Sample changes:")
for i in range(5):
    print(f"Original: {train_df['text'].iloc[i]}")
    print(f"Final: {train_df['final_text'].iloc[i]}")
    print"
```

```
Original vs Final:
Train original: 8000
Train final: 8000
Test original: 2000
Test final: 2000

Average lengths:
Original: 35.8
Final: 19.8

Sample changes:
Original: Our deeds are the reflection of our hearts.
Final: deed reflection heart

Original: Trust God with your life.
Final: trust God life

Original: Let us be kind to one another.
Final: kind one

Original: Let us be thankful for those who love us, and love them back.
Final: thankful love love back

Original: Let us be thankful for those who love us, and love them back.
Final: thankful love love back
```

Step 17 - Check results

```
%%capture
%%time
# Load the data
import pandas as pd
train_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_final.csv')
test_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_final.csv')

# Show statistics
print("Original vs Final:")
print("Train original:", train_df['text'].shape[0])
print("Train final:", train_df['final_text'].shape[0])
print("Test original:", test_df['text'].shape[0])
print("Test final:", test_df['final_text'].shape[0])

# Show average lengths
print\n"Average lengths:")
print("Original:", train_df['text'].str.length.mean())
print("Final:", train_df['final_text'].str.length.mean())

# Show some examples
print\n"Sample changes:")
for i in range(5):
    print(f"Original: {train_df['text'].iloc[i]}")
    print(f"Final: {train_df['final_text'].iloc[i]}")
    print"
```

```
Original vs Final:
Train original: 8000
Train final: 8000
Test original: 2000
Test final: 2000

Average lengths:
Original: 35.8
Final: 19.8

Sample changes:
Original: Our deeds are the reflection of our hearts.
Final: deed reflection heart

Original: Trust God with your life.
Final: trust God life

Original: Let us be kind to one another.
Final: kind one

Original: Let us be thankful for those who love us, and love them back.
Final: thankful love love back

Original: Let us be thankful for those who love us, and love them back.
Final: thankful love love back
```

Step 18 - Check results

```
%%capture
%%time
# Load the data
import pandas as pd
train_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_final.csv')
test_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_final.csv')

# Show statistics
print("Original vs Final:")
print("Train original:", train_df['text'].shape[0])
print("Train final:", train_df['final_text'].shape[0])
print("Test original:", test_df['text'].shape[0])
print("Test final:", test_df['final_text'].shape[0])

# Show average lengths
print\n"Average lengths:")
print("Original:", train_df['text'].str.length.mean())
print("Final:", train_df['final_text'].str.length.mean())

# Show some examples
print\n"Sample changes:")
for i in range(5):
    print(f"Original: {train_df['text'].iloc[i]}")
    print(f"Final: {train_df['final_text'].iloc[i]}")
    print"
```

```
Original vs Final:
Train original: 8000
Train final: 8000
Test original: 2000
Test final: 2000

Average lengths:
Original: 35.8
Final: 19.8

Sample changes:
Original: Our deeds are the reflection of our hearts.
Final: deed reflection heart

Original: Trust God with your life.
Final: trust God life

Original: Let us be kind to one another.
Final: kind one

Original: Let us be thankful for those who love us, and love them back.
Final: thankful love love back

Original: Let us be thankful for those who love us, and love them back.
Final: thankful love love back
```

Step 19 - Check results

```
%%capture
%%time
# Load the data
import pandas as pd
train_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_final.csv')
test_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_final.csv')

# Show statistics
print("Original vs Final:")
print("Train original:", train_df['text'].shape[0])
print("Train final:", train_df['final_text'].shape[0])
print("Test original:", test_df['text'].shape[0])
print("Test final:", test_df['final_text'].shape[0])

# Show average lengths
print\n"Average lengths:")
print("Original:", train_df['text'].str.length.mean())
print("Final:", train_df['final_text'].str.length.mean())

# Show some examples
print\n"Sample changes:")
for i in range(5):
    print(f"Original: {train_df['text'].iloc[i]}")
    print(f"Final: {train_df['final_text'].iloc[i]}")
    print"
```

```
Original vs Final:
Train original: 8000
Train final: 8000
Test original: 2000
Test final: 2000

Average lengths:
Original: 35.8
Final: 19.8

Sample changes:
Original: Our deeds are the reflection of our hearts.
Final: deed reflection heart

Original: Trust God with your life.
Final: trust God life

Original: Let us be kind to one another.
Final: kind one

Original: Let us be thankful for those who love us, and love them back.
Final: thankful love love back

Original: Let us be thankful for those who love us, and love them back.
Final: thankful love love back
```

Step 20 - Check results

```
%%capture
%%time
# Load the data
import pandas as pd
train_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/train_final.csv')
test_df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Christian_Text_Classification/data/processed/test_final.csv')

# Show statistics
print("Original vs Final:")
print("Train original:", train_df['text'].shape[0])
print("Train final:", train_df['final_text'].shape[0])
print("Test original:", test_df['text'].shape[0])
print("Test final:", test_df['final_text'].shape[0])

# Show average lengths
print\n"Average lengths:")
print("Original:", train_df['text'].str.length.mean())
print("Final:", train_df['final_text'].str.length.mean())

# Show some examples
print\n"Sample changes:")
for i in range(5):
    print(f"Original: {train_df['text'].iloc[i]}")
    print(f"Final: {train_df['final_text'].iloc[i]}")
    print"
```

```
Original vs Final:
Train original: 8000
Train final: 8000
Test original: 2000
Test final: 2000

Average lengths:
Original: 35.8
Final: 19.8

Sample changes:
Original: Our deeds are the reflection of our hearts.
Final: deed reflection heart

Original: Trust God with your life.
Final: trust God life

Original: Let us be kind to one another.
Final: kind one

Original: Let us be thankful for those who love us, and love them back.
Final: thankful love love back

Original: Let us be thankful for those who love us, and love them back.
Final: thankful love love back
```

Step 21 - Check results

```
%%capture
%%time
```

```
In [10]: from sklearn import feature_selection

         def cvtestscores(feature_selection_class,n_features,df,target,df):
             start = time.perf_counter()
             target = df[target].ravel()
             df = df.drop(target,axis=1)
             cv = StratifiedKFold(n_splits=10, random_state=1)
             lr = LogisticRegression(solver='liblinear',penalty='l1',C=1.0,class_weight='balanced')
             model = feature_selection_class(lr,n_features_to_select=n_features)
             # or remove all variances with zero variance from the dataset
             sel = VarianceThreshold(threshold=(0.0))
             df = sel.fit_transform(df)
             print('After removing zero variance features: %s' % df.shape)
             print('Selecting %d features' % n_features)
             scores = cross_val_score(model, df, target, scoring='roc_auc', cv=cv, n_jobs=-1)
             print('ROC AUC: %.3f (%.3f)' % (mean(scores), std(scores)))
             model = model.fit(df, target)
             print('Selected features: %s' % model.support_)
             end = time.perf_counter()
             print('Time to run feature selection and cross-validation: %s seconds' % (round(end - start)))
             cvtestscores_df = pd.DataFrame(data=[[str(feature_selection_class),n_features,round(mean(scores),3),round(
             cvtestscores_df.to_csv('data/cvtestscores.csv',mode='a',header=False)
             return model,df,target,cvtestscores_df

         cvtestscores(feature_selection.f_classif,'kbest',kbest_df,'target',kbest_df)
         cvtestscores(feature_selection.mutual_info_classif,'kbest',kbest_df,'target',kbest_df)
         cvtestscores(feature_selection.RFE,'rfe',rfe_df,'target',rfe_df)
         cvtestscores(feature_selection.SelectFromModel,'sfm',sfm_df,'target',sfm_df)
         cvtestscores(feature_selection.SelectPercentile,'percentile',percentile_df,'target',percentile_df)
         cvtestscores(feature_selection.SelectFpr,'fpr',fpr_df,'target',fpr_df)
         cvtestscores(feature_selection.SelectFdr,'fdr',fdr_df,'target',fdr_df)
         cvtestscores(feature_selection.SelectFwe,'fwe',fwe_df,'target',fwe_df)
         cvtestscores(feature_selection.GenericUnivariateSelect,'gus',gus_df,'target',gus_df)
```

```
In [11]: cvtestscores_df = pd.read_csv('data/cvtestscores.csv',header=None,
         cvtestscores_df.columns = ['feature_selection','n_features','roc_auc_score','time_to_run'
         cvtestscores_df.sort_values('roc_auc_score',ascending=False)
```

	feature_selection	n_features	roc_auc_score	time_to_run
0	sklearn.feature_selection.mutual_info_classif	kbest	0.900	1
1	sklearn.feature_selection.f_classif	kbest	0.900	1
2	sklearn.feature_selection.SelectFromModel	sfm	0.900	1
3	sklearn.feature_selection.SelectPercentile	percentile	0.900	1
4	sklearn.feature_selection.SelectFpr	fpr	0.900	1
5	sklearn.feature_selection.SelectFdr	fdr	0.900	1
6	sklearn.feature_selection.SelectFwe	fwe	0.900	1
7	sklearn.feature_selection.GenericUnivariateSelect	gus	0.900	1
8	sklearn.feature_selection.RFE	rfe	0.900	1

```
In [12]: kbest_df = kbest_df.drop('target',axis=1)
         rfe_df = rfe_df.drop('target',axis=1)
         sfm_df = sfm_df.drop('target',axis=1)
         percentile_df = percentile_df.drop('target',axis=1)
         fpr_df = fpr_df.drop('target',axis=1)
         fdr_df = fdr_df.drop('target',axis=1)
         fwe_df = fwe_df.drop('target',axis=1)
         gus_df = gus_df.drop('target',axis=1)
         rfe_df = rfe_df.drop('target',axis=1)
         kbest_df = kbest_df.ravel()
         rfe_df = rfe_df.ravel()
         sfm_df = sfm_df.ravel()
         percentile_df = percentile_df.ravel()
         fpr_df = fpr_df.ravel()
         fdr_df = fdr_df.ravel()
         fwe_df = fwe_df.ravel()
         gus_df = gus_df.ravel()
```

```
In [13]: cvtestscores(feature_selection.f_classif,'kbest',kbest_df,'target',kbest_df)
         cvtestscores(feature_selection.mutual_info_classif,'kbest',kbest_df,'target',kbest_df)
         cvtestscores(feature_selection.RFE,'rfe',rfe_df,'target',rfe_df)
         cvtestscores(feature_selection.SelectFromModel,'sfm',sfm_df,'target',sfm_df)
         cvtestscores(feature_selection.SelectPercentile,'percentile',percentile_df,'target',percentile_df)
         cvtestscores(feature_selection.SelectFpr,'fpr',fpr_df,'target',fpr_df)
         cvtestscores(feature_selection.SelectFdr,'fdr',fdr_df,'target',fdr_df)
         cvtestscores(feature_selection.SelectFwe,'fwe',fwe_df,'target',fwe_df)
         cvtestscores(feature_selection.GenericUnivariateSelect,'gus',gus_df,'target',gus_df)
```

```
In [14]: cvtestscores_df = pd.read_csv('data/cvtestscores.csv',header=None,
         cvtestscores_df.columns = ['feature_selection','n_features','roc_auc_score','time_to_run'
         cvtestscores_df.sort_values('roc_auc_score',ascending=False)
```

	feature_selection	n_features	roc_auc_score	time_to_run
0	sklearn.feature_selection.mutual_info_classif	kbest	0.900	1
1	sklearn.feature_selection.f_classif	kbest	0.900	1
2	sklearn.feature_selection.SelectFromModel	sfm	0.900	1
3	sklearn.feature_selection.SelectPercentile	percentile	0.900	1
4	sklearn.feature_selection.SelectFpr	fpr	0.900	1
5	sklearn.feature_selection.SelectFdr	fdr	0.900	1
6	sklearn.feature_selection.SelectFwe	fwe	0.900	1
7	sklearn.feature_selection.GenericUnivariateSelect	gus	0.900	1
8	sklearn.feature_selection.RFE	rfe	0.900	1

```
In [15]: kbest_df = kbest_df.drop('target',axis=1)
         rfe_df = rfe_df.drop('target',axis=1)
         sfm_df = sfm_df.drop('target',axis=1)
         percentile_df = percentile_df.drop('target',axis=1)
         fpr_df = fpr_df.drop('target',axis=1)
         fdr_df = fdr_df.drop('target',axis=1)
         fwe_df = fwe_df.drop('target',axis=1)
         gus_df = gus_df.drop('target',axis=1)
         rfe_df = rfe_df.drop('target',axis=1)
         kbest_df = kbest_df.ravel()
         rfe_df = rfe_df.ravel()
         sfm_df = sfm_df.ravel()
         percentile_df = percentile_df.ravel()
         fpr_df = fpr_df.ravel()
         fdr_df = fdr_df.ravel()
         fwe_df = fwe_df.ravel()
         gus_df = gus_df.ravel()
```

```
In [16]: cvtestscores(feature_selection.f_classif,'kbest',kbest_df,'target',kbest_df)
         cvtestscores(feature_selection.mutual_info_classif,'kbest',kbest_df,'target',kbest_df)
         cvtestscores(feature_selection.RFE,'rfe',rfe_df,'target',rfe_df)
         cvtestscores(feature_selection.SelectFromModel,'sfm',sfm_df,'target',sfm_df)
         cvtestscores(feature_selection.SelectPercentile,'percentile',percentile_df,'target',percentile_df)
         cvtestscores(feature_selection.SelectFpr,'fpr',fpr_df,'target',fpr_df)
         cvtestscores(feature_selection.SelectFdr,'fdr',fdr_df,'target',fdr_df)
         cvtestscores(feature_selection.SelectFwe,'fwe',fwe_df,'target',fwe_df)
         cvtestscores(feature_selection.GenericUnivariateSelect,'gus',gus_df,'target',gus_df)
```

```
In [17]: cvtestscores_df = pd.read_csv('data/cvtestscores.csv',header=None,
         cvtestscores_df.columns = ['feature_selection','n_features','roc_auc_score','time_to_run'
         cvtestscores_df.sort_values('roc_auc_score',ascending=False)
```

	feature_selection	n_features	roc_auc_score	time_to_run
0	sklearn.feature_selection.mutual_info_classif	kbest	0.900	1
1	sklearn.feature_selection.f_classif	kbest	0.900	1
2	sklearn.feature_selection.SelectFromModel	sfm	0.900	1
3	sklearn.feature_selection.SelectPercentile	percentile	0.900	1
4	sklearn.feature_selection.SelectFpr	fpr	0.900	1
5	sklearn.feature_selection.SelectFdr	fdr	0.900	1
6	sklearn.feature_selection.SelectFwe	fwe	0.900	1
7	sklearn.feature_selection.GenericUnivariateSelect	gus	0.900	1
8	sklearn.feature_selection.RFE	rfe	0.900	1

```
In [18]: kbest_df = kbest_df.drop('target',axis=1)
         rfe_df = rfe_df.drop('target',axis=1)
         sfm_df = sfm_df.drop('target',axis=1)
         percentile_df = percentile_df.drop('target',axis=1)
         fpr_df = fpr_df.drop('target',axis=1)
         fdr_df = fdr_df.drop('target',axis=1)
         fwe_df = fwe_df.drop('target',axis=1)
         gus_df = gus_df.drop('target',axis=1)
         rfe_df = rfe_df.drop('target',axis=1)
         kbest_df = kbest_df.ravel()
         rfe_df = rfe_df.ravel()
         sfm_df = sfm_df.ravel()
         percentile_df = percentile_df.ravel()
         fpr_df = fpr_df.ravel()
         fdr_df = fdr_df.ravel()
         fwe_df = fwe_df.ravel()
         gus_df = gus_df.ravel()
```

```
In [19]: cvtestscores(feature_selection.f_classif,'kbest',kbest_df,'target',kbest_df)
         cvtestscores(feature_selection.mutual_info_classif,'kbest',kbest_df,'target',kbest_df)
         cvtestscores(feature_selection.RFE,'rfe',rfe_df,'target',rfe_df)
         cvtestscores(feature_selection.SelectFromModel,'sfm',sfm_df,'target',sfm_df)
         cvtestscores(feature_selection.SelectPercentile,'percentile',percentile_df,'target',percentile_df)
         cvtestscores(feature_selection.SelectFpr,'fpr',fpr_df,'target',fpr_df)
         cvtestscores(feature_selection.SelectFdr,'fdr',fdr_df,'target',fdr_df)
         cvtestscores(feature_selection.SelectFwe,'fwe',fwe_df,'target',fwe_df)
         cvtestscores(feature_selection.GenericUnivariateSelect,'gus',gus_df,'target',gus_df)
```

```
In [20]: cvtestscores_df = pd.read_csv('data/cvtestscores.csv',header=None,
         cvtestscores_df.columns = ['feature_selection','n_features','roc_auc_score','time_to_run'
         cvtestscores_df.sort_values('roc_auc_score',ascending=False)
```

	feature_selection	n_features	roc_auc_score	time_to_run
0	sklearn.feature_selection.mutual_info_classif	kbest	0.900	1
1	sklearn.feature_selection.f_classif	kbest	0.900	1
2	sklearn.feature_selection.SelectFromModel	sfm	0.900	1
3	sklearn.feature_selection.SelectPercentile	percentile	0.900	1
4	sklearn.feature_selection.SelectFpr	fpr	0.900	1
5	sklearn.feature_selection.SelectFdr	fdr	0.900	1
6	sklearn.feature_selection.SelectFwe	fwe	0.900	1
7	sklearn.feature_selection.GenericUnivariateSelect	gus	0.900	1
8	sklearn.feature_selection.RFE	rfe	0.900	1

```
In [21]: kbest_df = kbest_df.drop('target',axis=1)
         rfe_df = rfe_df.drop('target',axis=1)
         sfm_df = sfm_df.drop('target',axis=1)
         percentile_df = percentile_df.drop('target',axis=1)
         fpr_df = fpr_df.drop('target',axis=1)
         fdr_df = fdr_df.drop('target',axis=1)
         fwe_df = fwe_df.drop('target',axis=1)
         gus_df = gus_df.drop('target',axis=1)
         rfe_df = rfe_df.drop('target',axis=1)
         kbest_df = kbest_df.ravel()
         rfe_df = rfe_df.ravel()
         sfm_df = sfm_df.ravel()
         percentile_df = percentile_df.ravel()
         fpr_df = fpr_df.ravel()
         fdr_df = fdr_df.ravel()
         fwe_df = fwe_df.ravel()
         gus_df = gus_df.ravel()
```

```
In [22]: cvtestscores(feature_selection.f_classif,'kbest',kbest_df,'target',kbest_df)
         cvtestscores(feature_selection.mutual_info_classif,'kbest',kbest_df,'target',kbest_df)
         cvtestscores(feature_selection.RFE,'rfe',rfe_df,'target',rfe_df)
         cvtestscores(feature_selection.SelectFromModel,'sfm',sfm_df,'target',sfm_df)
         cvtestscores(feature_selection.SelectPercentile,'percentile',percentile_df,'target',percentile_df)
         cvtestscores(feature_selection.SelectFpr,'fpr',fpr_df,'target',fpr_df)
         cvtestscores(feature_selection.SelectFdr,'fdr',fdr_df,'target',fdr_df)
         cvtestscores(feature_selection.SelectFwe,'fwe',fwe_df,'target',fwe_df)
         cvtestscores(feature_selection.GenericUnivariateSelect,'gus',gus_df,'target',gus_df)
```

```
In [23]: cvtestscores_df = pd.read_csv('data/cvtestscores.csv',header=None,
         cvtestscores_df.columns = ['feature_selection','n_features','roc_auc_score','time_to_run'
         cvtestscores_df.sort_values('roc_auc_score',ascending=False)
```

	feature_selection	n_features	roc_auc_score	time_to_run
0	sklearn.feature_selection.mutual_info_classif	kbest	0.900	1
1	sklearn.feature_selection.f_classif	kbest	0.900	1
2	sklearn.feature_selection.SelectFromModel	sfm	0.900	1
3	sklearn.feature_selection.SelectPercentile	percentile	0.900	1
4	sklearn.feature_selection.SelectFpr	fpr	0.900	1
5	sklearn.feature_selection.SelectFdr	fdr	0.900	1
6	sklearn.feature_selection.SelectFwe	fwe	0.900	1
7	sklearn.feature_selection.GenericUnivariateSelect	gus	0.900	1
8	sklearn.feature_selection.RFE	rfe	0.900	1

```
In [24]: kbest_df = kbest_df.drop('target',axis=1)
         rfe_df = rfe_df.drop('target',axis=1)
         sfm_df = sfm_df.drop('target',axis=1)
         percentile_df = percentile_df.drop('target',axis=1)
         fpr_df = fpr_df.drop('target',axis=1)
         fdr_df = fdr_df.drop('target',axis=1)
         fwe_df = fwe_df.drop('target',axis=1)
         gus_df = gus_df.drop('target',axis=1)
         rfe_df = rfe_df.drop('target',axis=1)
         kbest_df = kbest_df.ravel()
         rfe_df = rfe_df.ravel()
         sfm_df = sfm_df.ravel()
         percentile_df = percentile_df.ravel()
         fpr_df = fpr_df.ravel()
         fdr_df = fdr_df.ravel()
         fwe_df = fwe_df.ravel()
         gus_df = gus_df.ravel()
```

```
In [25]: cvtestscores(feature_selection.f_classif,'kbest',kbest_df,'target',kbest_df)
         cvtestscores(feature_selection.mutual_info_classif,'kbest',kbest_df,'target',kbest_df)
         cvtestscores(feature_selection.RFE,'rfe',rfe_df,'target',rfe_df)
         cvtestscores(feature_selection.SelectFromModel,'sfm',sfm_df,'target',sfm_df)
         cvtestscores(feature_selection.SelectPercentile,'percentile',percentile_df,'target',percentile_df)
         cvtestscores(feature_selection.SelectFpr,'fpr',fpr_df,'target',fpr_df)
         cvtestscores(feature_selection.SelectFdr,'fdr',fdr_df,'target',fdr_df)
         cvtestscores(feature_selection.SelectFwe,'fwe',fwe_df,'target',fwe_df)
         cvtestscores(feature_selection.GenericUnivariateSelect,'gus',gus_df,'target',gus_df)
```

```
In [26]: cvtestscores_df = pd.read_csv('data/cvtestscores.csv',header=None,
         cvtestscores_df.columns = ['feature_selection','n_features','roc_auc_score','time_to_run'
         cvtestscores_df.sort_values('roc_auc_score',ascending=False)
```

	feature_selection	n_features	roc_auc_score	time_to_run
0	sklearn.feature_selection.mutual_info_classif	kbest	0.900	1
1	sklearn.feature_selection.f_classif	kbest	0.900	1
2	sklearn.feature_selection.SelectFromModel	sfm	0.900	1
3	sklearn.feature_selection.SelectPercentile	percentile	0.900	1
4	sklearn.feature_selection.SelectFpr	fpr	0.900	1
5	sklearn.feature_selection.SelectFdr	fdr	0.900	1
6	sklearn.feature_selection.SelectFwe	fwe	0.900	1
7	sklearn.feature_selection.GenericUnivariateSelect	gus	0.900	1
8	sklearn.feature_selection.RFE	rfe	0.900	1

```
In [27]: kbest_df = kbest_df.drop('target',axis=1)
         rfe_df = rfe_df.drop('target',axis=1)
         sfm_df = sfm_df.drop('target',axis=1)
         percentile_df = percentile_df.drop('target',axis=1)
         fpr_df = fpr_df.drop('target',axis=1)
         fdr_df = fdr_df.drop('target',axis=1)
         fwe_df = fwe_df.drop('target',axis=1)
         gus_df = gus_df.drop('target',axis=1)
         rfe_df = rfe_df.drop('target',axis=1)
         kbest_df = kbest_df.ravel()
         rfe_df = rfe_df.ravel()
         sfm_df = sfm_df.ravel()
         percentile_df = percentile_df.ravel()
         fpr_df = fpr_df.ravel()
         fdr_df = fdr_df.ravel()
         fwe_df = fwe_df.ravel()
         gus_df = gus_df.ravel()
```

```
In [28]: cvtestscores(feature_selection.f_classif,'kbest',kbest_df,'target',kbest_df)
         cvtestscores(feature_selection.mutual_info_classif,'kbest',kbest_df,'target',kbest_df)
         cvtestscores(feature_selection.RFE,'rfe',rfe_df,'target',rfe_df)
         cvtestscores(feature_selection.SelectFromModel,'sfm',sfm_df,'target',sfm_df)
         cvtestscores(feature_selection.SelectPercentile,'percentile',percentile_df,'target',percentile_df)
         cvtestscores(feature_selection.SelectFpr,'fpr',fpr_df,'target',fpr_df)
         cvtestscores(feature_selection.SelectFdr,'fdr',fdr_df,'target',fdr_df)
         cvtestscores(feature_selment.SelectFwe,'fwe',fwe_df,'target',fwe_df)
         cvtestscores(feature_selection.GenericUnivariateSelect,'gus',gus_df,'target',gus_df)
```

```
In [29]: cvtestscores_df = pd.read_csv('data/cvtestscores.csv',header=None,
         cvtestscores_df.columns = ['feature_selection','n_features','roc_auc_score','time_to_run'
         cvtestscores_df.sort_values('roc_auc_score',ascending=False)
```

	feature_selection	n_features	roc_auc_score	time_to_run
0	sklearn.feature_selection.mutual_info_classif	kbest	0.900	1
1	sklearn.feature_selection.f_classif	kbest	0.900	1
2	sklearn.feature_selection.SelectFromModel	sfm	0.900	1
3	sklearn.feature_selection.SelectPercentile	percentile	0.900	1
4	sklearn.feature_selection.SelectFpr	fpr	0.900	1
5	sklearn.feature_selection.SelectFdr	fdr	0.900	1
6	sklearn.feature_selment.SelectFwe	fwe	0.900	1
7	sklearn.feature_selection.GenericUnivariateSelect	gus	0.900	1
8	sklearn.feature_selection.RFE	rfe	0.900	1

```
In [30]: kbest_df = kbest_df.drop('target',axis=1)
         rfe_df = rfe_df.drop('target',axis=1)
         sfm_df = sfm_df.drop('target',axis=1)
         percentile_df = percentile_df.drop('target',axis=1)
         fpr_df = fpr_df.drop('target',axis=1)
         fdr_df = fdr_df.drop('target',axis=1)
         fwe_df = fwe_df.drop('target',axis=1)
         gus_df = gus_df.drop('target',axis=1)
         rfe_df = rfe_df.drop('target',axis=1)
         kbest_df = kbest_df.ravel()
         rfe_df = rfe_df.ravel()
         sfm_df = sfm_df.ravel()
         percentile_df = percentile_df.ravel()
         fpr_df = fpr_df.ravel()
         fdr_df = fdr_df.ravel()
         fwe_df = fwe_df.ravel()
         gus_df = gus_df.ravel()
```

```
In [31]: cvtestscores(feature_selection.f_classif,'kbest',kbest_df,'target',kbest_df)
         cvtestscores(feature_selection.mutual_info_classif,'kbest',kbest_df,'target',kbest_df)
         cvtestscores(feature_selection.RFE,'rfe',rfe_df,'target',rfe_df)
         cvtestscores(feature_selection.SelectFromModel,'sfm',sfm_df,'target',sfm_df)
         cvtestscores(feature_selection.SelectPercentile,'percentile',percentile_df,'target',percentile_df)
         cvtestscores(feature_selection.SelectFpr,'fpr',fpr_df,'target',fpr_df)
         cvtestscores(feature_selection.SelectFdr,'fdr',fdr_df,'target',fdr_df)
         cvtestscores(feature_selment.SelectFwe,'fwe',fwe_df,'target',fwe_df)
         cvtestscores(feature_selection.GenericUnivariateSelect,'gus',gus_df,'target',gus_df)
```

```
In [32]: cvtestscores_df = pd.read_csv('data/cvtestscores.csv',header=None,
         cvtestscores_df.columns = ['feature_selection','n_features','roc_auc_score','time_to_run'
         cvtestscores_df.sort_values('roc_auc_score',ascending=False)
```

	feature_selection	n_features	roc_auc_score	time_to_run
0	sklearn.feature_selection.mutual_info_classif	kbest	0.900	1
1	sklearn.feature_selection.f_classif	kbest	0.900	1
2	sklearn.feature_selection.SelectFromModel	sfm	0.900	1
3	sklearn.feature_selection.SelectPercentile	percentile	0.900	1
4	sklearn.feature_selection.SelectFpr	fpr	0.900	1
5	sklearn.feature_selection.SelectFdr	fdr	0.900	1
6	sklearn.feature_selment.SelectFwe	fwe	0.900	1
7	sklearn.feature_selection.GenericUnivariateSelect	gus	0.900	1
8	sklearn.feature_selection.RFE	rfe	0.900	1

```
In [33]: kbest_df = kbest_df.drop('target',axis=1)
         rfe_df = rfe_df.drop('target',axis=1)
         sfm_df = sfm_df.drop('target',axis=1)
         percentile_df = percentile_df.drop('target',axis=1)
         fpr_df = fpr_df.drop('target',axis=1)
         fdr_df = fdr_df.drop('target',axis=1)
         fwe_df = fwe_df.drop('target',axis=1)
         gus_df = gus_df.drop('target',axis=1)
         rfe_df = rfe_df.drop('target',axis=1)
         kbest_df = kbest_df.ravel()
         rfe_df = rfe_df.ravel()
         sfm_df = sfm_df.ravel()
         percentile_df = percentile_df.ravel()
         fpr_df = fpr_df.ravel()
         fdr_df = fdr_df.ravel()
         fwe_df = fwe_df.ravel()
         gus_df = gus_df.ravel()
```

```
In [34]: cvtestscores(feature_selection.f_classif,'kbest',kbest_df,'target',kbest_df)
         cvtestscores(feature_selection.mutual_info_classif,'kbest',kbest_df,'target',kbest_df)
         cvtestscores(feature_selection.RFE,'rfe',rfe_df,'target',rfe_df)
         cvtestscores(feature_selection.SelectFromModel,'sfm',sfm_df,'target',sfm_df)
         cvtestscores(feature_selection.SelectPercentile,'percentile',percentile_df,'target',percentile_df)
         cvtestscores(feature_selection.SelectFpr,'fpr',fpr_df,'target',fpr_df)
         cvtestscores(feature_selection.SelectFdr,'fdr',fdr_df,'target',fdr_df)
         cvtestscores(feature_selment.SelectFwe,'fwe',fwe_df,'target',fwe_df)
         cvtestscores(feature_selection.GenericUnivariateSelect,'gus',gus_df,'target',gus_df)
```

```
In [35]: cvtestscores_df = pd.read_csv('data/cvtestscores.csv',header=None,
         cvtestscores_df.columns = ['feature_selection','n_features','roc_auc_score','time_to_run'
         cvtestscores_df.sort_values('roc_auc_score',ascending=False)
```

	feature_selection	n_features	roc_auc_score	time_to_run
0	sklearn.feature_selection.mutual_info_classif	kbest	0.900	1
1	sklearn.feature_selection.f_classif	kbest	0.900	1
2	sklearn.feature_selection.SelectFromModel	sfm	0.900	1
3	sklearn.feature_selection.SelectPercentile	percentile	0.900	1
4	sklearn.feature_selection.SelectFpr	fpr	0.900	1
5	sklearn.feature_selection.SelectFdr	fdr	0.900	1
6	sklearn.feature_selment.SelectFwe	fwe	0.900	1
7	sklearn.feature_selection.GenericUnivariateSelect	gus	0.900	1
8	sklearn.feature_selection.RFE	rfe	0.900	1

```
In [36]: kbest_df = kbest_df.drop('target',axis=1)
         rfe_df = rfe_df.drop('target',axis=1)
         sfm_df = sfm_df.drop('target',axis=1)
         percentile_df = percentile_df.drop('target',axis=1)
         fpr_df = fpr_df.drop('target',axis=1)
         fdr_df = fdr_df.drop('target',axis=1)
         fwe_df = fwe_df.drop('target',axis=1)
         gus_df = gus_df.drop('target',axis=1)
         rfe_df = rfe_df.drop('target',axis=1)
         kbest_df = kbest_df.ravel()
         rfe_df = rfe_df.ravel()
         sfm_df = sfm_df.ravel()
         percentile_df = percentile_df.ravel()
         fpr_df = fpr_df.ravel()
         fdr_df = fdr_df.ravel()
         fwe_df = fwe_df.ravel()
         gus_df = gus_df.ravel()
```

```
In [37]: cvtestscores(feature_selection.f_classif,'kbest',kbest_df,'target',kbest_df)
         cvtestscores(feature_selection.mutual_info_classif,'kbest',kbest_df,'target',kbest_df)
         cvtestscores(feature_selection.RFE,'rfe',rfe_df,'target',rfe_df)
         cvtestscores(feature_selection.SelectFromModel,'sfm',sfm_df,'target',sfm_df)
         cvtestscores(feature_selection.SelectPercentile,'percentile',percentile_df,'target',percentile_df)
         cvtestscores(feature_selection.SelectFpr,'fpr',fpr_df,'target',fpr_df)
         cvtestscores(feature_selection.SelectFdr,'fdr',fdr_df,'target',fdr_df)
         cvtestscores(feature_selment.SelectFwe,'fwe',fwe_df,'target',fwe_df)
         cvtestscores(feature_selection.GenericUnivariateSelect,'gus',gus_df,'target',gus_df)
```

```
In [38]: cvtestscores_df = pd.read_csv('data/cvtestscores.csv',header=None,
         cvtestscores_df.columns = ['feature_selection','n_features','roc_auc_score','time_to_run'
         cvtestscores_df.sort_values('roc_auc_score',ascending=False)
```

	feature_selection	n_features	roc_auc_score	time_to_run
0	sklearn.feature_selection.mutual_info_classif	kbest	0.900	1
1	sklearn.feature_selection.f_classif	kbest	0.900	1
2	sklearn.feature_selection.SelectFromModel	sfm	0.900	1
3	sklearn.feature_selection.SelectPercentile	percentile	0.900	1
4	sklearn.feature_selection.SelectFpr	fpr	0.900	1
5	sklearn.feature_selection.SelectFdr	fdr	0.900	1
6	sklearn.feature_selment.SelectFwe	fwe	0.900	1
7	sklearn.feature_selection.GenericUnivariateSelect	gus	0.900	1
8	sklearn.feature_selection.RFE	rfe	0.900	1

```
In [39]: kbest_df = kbest_df.drop('target',axis=1)
         rfe_df = rfe_df.drop('target',axis=1)
         sfm_df = sfm_df.drop('target',axis=1)
         percentile_df = percentile_df.drop('target',axis=1)
         fpr_df = fpr_df.drop('target',axis=1)
         fdr_df = fdr_df.drop('target',axis=1)
         fwe_df = fwe_df.drop('target',axis=1)
         gus_df = gus_df.drop('target',axis=1)
         rfe_df = rfe_df.drop('target',axis=1)
         kbest_df = kbest_df.ravel()
         rfe_df = rfe_df.ravel()
         sfm_df = sfm_df.ravel()
         percentile_df = percentile_df.ravel()
         fpr_df = fpr_df.ravel()
         fdr_df = fdr_df.ravel()
         fwe_df = fwe_df.ravel()
         gus_df = gus_df.ravel()
```

```
In [40]: cvtestscores(feature_selection.f_classif,'kbest',kbest_df,'target',kbest_df)
         cvtestscores(feature_selection.mutual_info_classif,'kbest',kbest_df,'target',kbest_df)
         cvtestscores(feature_selection.RFE,'rfe',rfe_df,'target',rfe_df)
         cvtestscores(feature_selection.SelectFromModel,'sfm',sfm_df,'target',sfm_df)
         cvtestscores(feature_selection.SelectPercentile,'percentile',percentile_df,'target',percentile_df)
         cvtestscores(feature_selection.SelectFpr,'fpr',fpr_df,'target',fpr_df)
         cvtestscores(feature_selection.SelectFdr,'fdr',fdr_df,'target',fdr_df)
         cvtestscores(feature_selment.SelectFwe,'fwe',fwe_df,'target',fwe_df)
         cvtestscores(feature_selection.GenericUnivariateSelect,'gus',gus_df,'target',gus_df)
```

```
In [41]: cvtestscores_df = pd.read_csv('data/cvtestscores.csv',header=None,
         cvtestscores_df.columns = ['feature_selection','n_features','roc_auc_score','time_to_run'
         cvtestscores_df.sort_values('roc_auc_score',ascending=False)
```

	feature_selection	n_features	roc_auc_score	time_to_run
0	sklearn.feature_selection.mutual_info_classif	kbest	0.900	1
1	sklearn.feature_selection.f_classif	kbest	0.900	1
2	sklearn.feature_selection.SelectFromModel	sfm	0.900	1
3	sklearn.feature_selection.SelectPercentile	percentile	0.900	1
4	sklearn.feature_selection.SelectFpr	fpr	0.900	1
5	sklearn.feature_selection.SelectFdr	fdr	0.900	1
6	sklearn.feature_selment.SelectFwe	fwe	0.900	1
7	sklearn.feature_selection.GenericUnivariateSelect	gus	0.900	1
8	sklearn.feature_selection.RFE	rfe	0.900	1

```
In [42]: kbest_df = kbest_df.drop('target',axis=1)
         rfe_df = rfe_df.drop('target',axis=1)
         sfm_df = sfm_df.drop('target',axis=1)
         percentile_df = percentile_df.drop('target',axis=1)
         fpr_df = fpr_df.drop('target',axis=1)
         fdr_df = fdr_df.drop('target',axis=1)
         fwe_df = fwe_df.drop('target',axis=1)
         gus_df = gus_df.drop('target',axis=1)
         rfe_df = rfe_df.drop('target',axis=1)
         kbest_df = kbest_df.ravel()
         rfe_df = rfe_df.ravel()
         sfm_df = sfm_df.ravel()
         percentile_df = percentile_df.ravel()
         fpr_df = fpr_df.ravel()
         fdr_df = fdr_df.ravel()
         fwe_df = fwe_df.ravel()
         gus_df = gus_df.ravel()
```

```
In [43]: cvtestscores(feature_selection.f_classif,'kbest',kbest_df,'target',kbest_df)
         cvtestscores(feature_selection.mutual_info_classif,'kbest',kbest_df,'target',kbest_df)
         cvtestscores(feature_selection.RFE,'rfe',rfe_df,'target',rfe_df)
         cvtestscores(feature_selection.SelectFromModel,'sfm',sfm_df,'target',sfm_df)
         cvtestscores(feature_selection.SelectPercentile,'percentile',percentile_df,'target',percentile_df)
         cvtestscores(feature_selection.SelectFpr,'fpr',fpr_df,'target',fpr_df)
         cvtestscores(feature_selection.SelectFdr,'fdr',fdr_df,'target',fdr_df)
         cvtestscores(feature_selment.SelectFwe,'fwe',fwe_df,'target',fwe_df)
         cvtestscores(feature_selection.GenericUnivariateSelect,'gus',gus_df,'target',gus_df)
```

```
In [44]: cvtestscores_df = pd.read_csv('data/cvtestscores.csv',header=None,
         cvtestscores_df.columns = ['feature_selection','n_features','roc_auc_score','time_to_run'
         cvtestscores_df.sort_values('roc_auc_score',ascending=False)
```

	feature_selection	n_features	roc_auc_score	time_to_run
0	sklearn.feature_selection.mutual_info_classif	kbest	0.900	1
1	sklearn.feature_selection.f_classif	kbest	0.900	1
2	sklearn.feature_selection.SelectFromModel	sfm	0.900	1
3	sklearn.feature_selection.SelectPercentile	percentile	0.900	1
4	sklearn.feature_selection.SelectFpr	fpr	0.900	1
5	sklearn.feature_selection.SelectFdr	fdr	0.900	1
6	sklearn.feature_selment.SelectFwe	fwe	0.900	1
7	sklearn.feature_selection.GenericUnivariateSelect	gus	0.900	1
8	sklearn.feature_selection.RFE	rfe	0.900	1

```
In [45]: kbest_df = kbest_df.drop('target',axis=1)
         rfe_df = rfe_df.drop('target',axis=1)
         sfm_df = sfm_df.drop('target',axis=1)
         percentile_df = percentile_df.drop('target',axis=1)
         fpr_df = fpr_df.drop('target',axis=1)
         fdr_df = fdr_df.drop('target',axis=1)
         fwe_df = fwe_df.drop('target',axis=1)
         gus_df = gus_df.drop('target',axis=1)
         rfe_df = rfe_df.drop('target',axis=1)
         kbest_df = kbest_df.ravel()
         rfe_df = rfe_df.ravel()
         sfm_df = sfm_df.ravel()
         percentile_df = percentile_df.ravel()
         fpr_df = fpr_df.ravel()
         fdr_df = fdr_df.ravel()
         fwe_df = fwe_df.ravel()
         gus_df = gus_df.ravel()
```

```
In [46]: cvtestscores(feature_selection.f_classif,'kbest',kbest_df,'target',kbest_df)
         cvtestscores(feature_selection.mutual_info_classif,'kbest',kbest_df,'target',kbest_df)
         cvtestscores(feature_selection.RFE,'rfe',rfe_df,'target',rfe_df)
         cvtestscores(feature_selection.SelectFromModel,'sfm',sfm_df,'target',sfm_df)
         cvtestscores(feature_selection.SelectPercentile,'percentile',percentile_df,'target',percentile_df)
         cvtestscores(feature_selection.SelectFpr,'fpr',fpr_df,'target',fpr_df)
         cvtestscores(feature_selection.SelectFdr,'fdr',fdr_df,'target',fdr_df)
         cvtestscores(feature_selment.SelectFwe,'fwe',fwe_df,'target',fwe_df)
         cvtestscores(feature_selection.GenericUnivariateSelect,'gus',gus_df,'target',gus_df)
```

```
In [47]: cvtestscores_df = pd.read_csv('data/cvtestscores.csv',header=None,
         cvtestscores_df.columns = ['feature_selection','n_features','roc_auc_score','time_to_run'
         cvtestscores_df.
```