```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.feature_selection import chi2, f_classif, mutual_info_classif
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectKBest
from datetime import datetime
import seaborn as sns
import matplotlib.pyplot as plt
```

```
df = pd.DataFrame({
    'age': [25, 32, 47, 51, 62, 41, 29, 35, 58, 48],
    'salary': [40000, 52000, 75000, 81000, 120000, 72000, 48000, 60000, 100000, 90000],
    'gender': ['F', 'M', 'M', 'M', 'F', 'M', 'F', 'F', 'M', 'F'],
    'joining_date': ['2021-05-12', '2019-03-25', '2018-08-14', '2017-11-01', '2015-01-20',
                     '2020-06-30', '2019-07-22', '2021-12-15', '2016-02-11', '2018-10-05'],
    'review_text': [
        'Excellent work and dedication',
        'Good performance overall',
        'Needs improvement in communication',
        'Outstanding leadership',
        'Average contribution',
        'Great team player',
        'Punctual and consistent',
        'Highly skilled and proactive',
        'Lacks creativity sometimes',
        'Very dependable and trustworthy'
    ],
    'performance': [1, 1, 0, 1, 0, 1, 1, 1, 0, 1]  # Target variable
})

df['joining_date'] = pd.to_datetime(df['joining_date'])
df.head()
```

	age	salary	gender	joining_date	review_text	performance
0	25	40000	F	2021-05-12	Excellent work and dedication	1
1	32	52000	M	2019-03-25	Good performance overall	1
2	47	75000	M	2018-08-14	Needs improvement in communication	0
3	51	81000	M	2017-11-01	Outstanding leadership	1
4	62	120000	F	2015-01-20	Average contribution	0

```
#statistical features
df['salary_mean'] = df['salary'].mean()
df['salary_std'] = df['salary'].std()
df['salary_zscore'] = (df['salary'] - df['salary_mean']) / df['salary_std']

print(df[['salary', 'salary_mean', 'salary_std', 'salary_zscore']].head())
```

```
   salary  salary_mean    salary_std  salary_zscore
4  120000      73800.0  24974.653818       1.849875
8  100000      73800.0  24974.653818       1.049064
3   81000      73800.0  24974.653818       0.288292
2   75000      73800.0  24974.653818       0.048049
9   90000      73800.0  24974.653818       0.648658
```

```
# Label Encoding
le = LabelEncoder()
df['gender_encoded'] = le.fit_transform(df['gender'])

# One-hot encoding
df = pd.get_dummies(df, columns=['gender'], drop_first=True)
```

```
df['joining_year'] = df['joining_date'].dt.year
df['joining_month'] = df['joining_date'].dt.month
df['joining_day'] = df['joining_date'].dt.day
df['years_since_joining'] = datetime.now().year - df['joining_year']

df[['joining_date', 'joining_year', 'joining_month', 'years_since_joining']].head()
```

	joining_date	joining_year	joining_month	years_since_joining
0	2021-05-12	2021	5	4
1	2019-03-25	2019	3	6
2	2018-08-14	2018	8	7
3	2017-11-01	2017	11	8
4	2015-01-20	2015	1	10

```
# Sorting by joining date
df = df.sort_values('joining_date')
```

```
# Lag feature for salary (previous row)
df['salary_lag1'] = df['salary'].shift(1)
df['salary_diff'] = df['salary'] - df['salary_lag1']

df[['salary', 'salary_lag1', 'salary_diff']].head()
```

	salary	salary_lag1	salary_diff
4	120000	NaN	NaN
8	100000	120000.0	-20000.0
3	81000	100000.0	-19000.0
2	75000	81000.0	-6000.0
9	90000	75000.0	15000.0

```
# Text-based feature engineering
df['text_length'] = df['review_text'].apply(len)
df['word_count'] = df['review_text'].apply(lambda x: len(x.split()))
df['avg_word_length'] = df['review_text'].apply(lambda x: np.mean([len(word) for word in x.split()]))

df[['review_text', 'text_length', 'word_count', 'avg_word_length']].head()
```

	review_text	text_length	word_count	avg_word_length
4	Average contribution	20	2	9.50
8	Lacks creativity sometimes	26	3	8.00
3	Outstanding leadership	22	2	10.50
2	Needs improvement in communication	34	4	7.75
9	Very dependable and trustworthy	31	4	7.00

Feature Selection

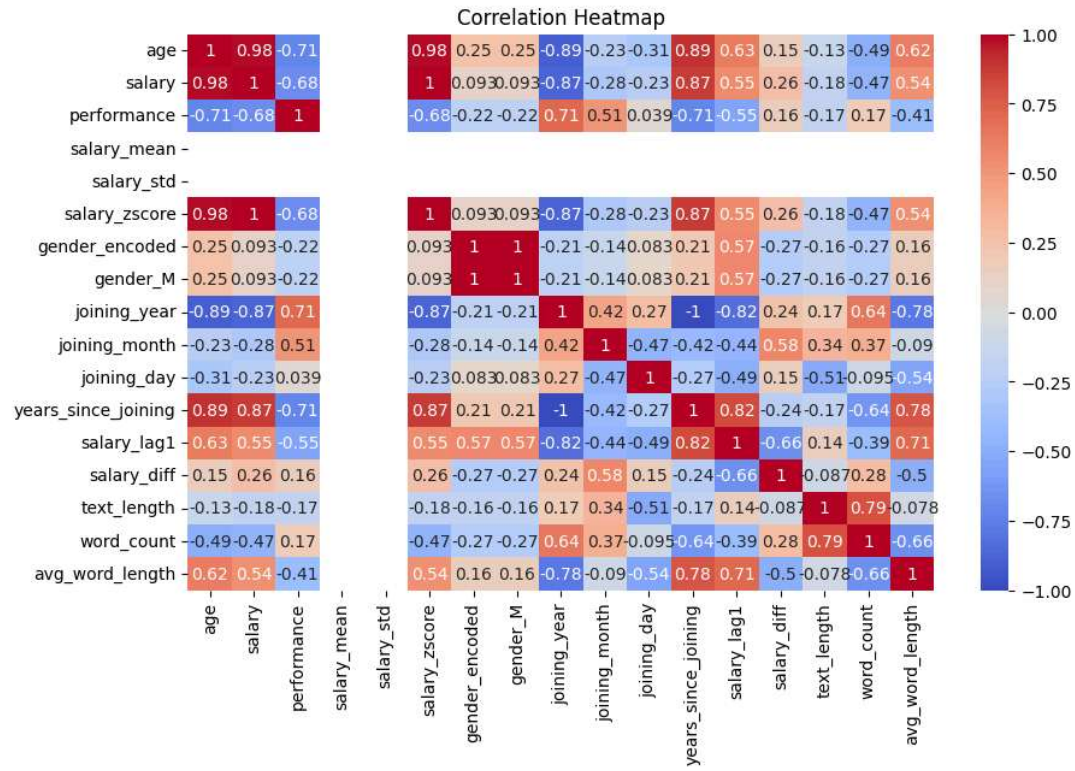
```
X = df.select_dtypes(include=[np.number]).drop(columns=['performance'])  # all numeric features
y = df['performance']

X.head()
```

	age	salary	salary_mean	salary_std	salary_zscore	gender_encoded	joining_year	joining_month	joining_day	years_since_joining	salary_
4	62	120000	73800.0	24974.653818	1.849875	0	2015	1	20	10	
8	58	100000	73800.0	24974.653818	1.049064	1	2016	2	11	9	120
3	51	81000	73800.0	24974.653818	0.288292	1	2017	11	1	8	100
2	47	75000	73800.0	24974.653818	0.048049	1	2018	8	14	7	81
9	48	90000	73800.0	24974.653818	0.648658	0	2018	10	5	7	75

```
corr_matrix = df.corr(numeric_only=True)
plt.figure(figsize=(10, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()

# Finding features highly correlated with target
corr_target = corr_matrix['performance'].sort_values(ascending=False)
print("\nFeatures correlated with 'performance':\n", corr_target)
```




```
Features correlated with 'performance':
 performance            1.000000
joining_year            0.709139
joining_month           0.513457
word_count              0.174964
salary_diff             0.158958
joining_day             0.038616
text_length            -0.167735
gender_encoded         -0.218218
gender_M               -0.218218
avg_word_length        -0.405989
salary_lag1            -0.549080
salary_zscore          -0.677872
salary                 -0.677872
years_since_joining    -0.709139
age                    -0.711993
salary_mean                  NaN
salary_std                   NaN
Name: performance, dtype: float64
```

```
from sklearn.feature_selection import chi2

# Filling missing values
X_chi2 = X.fillna(0).copy()

# Chi-square requires all features to be non-negative
# So we drop any columns that have negative values
X_chi2 = X_chi2.loc[:, (X_chi2 >= 0).all()]

# removing continuous difference columns
if 'salary_diff' in X_chi2.columns:
    X_chi2 = X_chi2.drop(columns=['salary_diff'])

# Appling Chi-square test
chi_scores, p_values = chi2(X_chi2, y)

# Creating DataFrame for results
chi2_results = pd.DataFrame({
    'Feature': X_chi2.columns,
    'Chi2 Score': chi_scores,
    'p-value': p_values
}).sort_values('Chi2 Score', ascending=False)

print("\n Chi-square Test Results (non-negative features only):\n")
print(chi2_results)
```

```
 Chi-square Test Results (non-negative features only):

                Feature    Chi2 Score       p-value
1                salary  3.495264e+04  0.000000e+00
9           salary_lag1  4.045512e+01  2.011845e-10
0                   age  1.657721e+01  4.670885e-05
6         joining_month  5.293040e+00  2.141081e-02
8   years_since_joining  2.773449e+00  9.583947e-02
12      avg_word_length  4.988864e-01  4.799898e-01
10          text_length  2.707162e-01  6.028518e-01
4         gender_encoded  2.380952e-01  6.255852e-01
```

```
7           joining_day  6.912442e-02  7.926159e-01
11           word_count  5.357143e-02  8.169613e-01
5          joining_year  9.068947e-03  9.241313e-01
3            salary_std  7.671409e-27  1.000000e+00
2           salary_mean  6.558521e-27  1.000000e+00
```

```
#Anova f-test
f_scores, p_values = f_classif(X.fillna(0), y)
anova_results = pd.DataFrame({'Feature': X.columns, 'F-Score': f_scores, 'p-value': p_values})
anova_results = anova_results.sort_values('F-Score', ascending=False)
print("\nANOVA F-test Results:\n", anova_results)
```

```
ANOVA F-test Results:
                Feature   F-Score   p-value
0                   age  8.225007  0.020895
9   years_since_joining  8.092632  0.021653
6          joining_year  8.092632  0.021653
4         salary_zscore  6.801381  0.031230
1                salary  6.801381  0.031230
7         joining_month  2.864222  0.129028
14      avg_word_length  1.578855  0.244374
5        gender_encoded  0.400000  0.544737
13           word_count  0.252632  0.628770
12          text_length  0.231596  0.643230
11          salary_diff  0.059589  0.813291
8           joining_day  0.011947  0.915653
10          salary_lag1  0.002088  0.964670
3            salary_std      -inf       NaN
2           salary_mean       NaN       NaN
/usr/local/lib/python3.12/dist-packages/sklearn/feature_selection/_univariate_selection.py:111: UserWarning: Features [2 3] are constant.
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
/usr/local/lib/python3.12/dist-packages/sklearn/feature_selection/_univariate_selection.py:112: RuntimeWarning: divide by zero encountered in
  f = msb / msw
/usr/local/lib/python3.12/dist-packages/sklearn/feature_selection/_univariate_selection.py:112: RuntimeWarning: invalid value encountered in d
  f = msb / msw
```

```
#mutual information
mi = mutual_info_classif(X.fillna(0), y)
mi_results = pd.DataFrame({'Feature': X.columns, 'Mutual Information': mi})
mi_results = mi_results.sort_values('Mutual Information', ascending=False)
print("\nMutual Information Results:\n", mi_results)
```

```
Mutual Information Results:
                Feature  Mutual Information
2           salary_mean            1.448175
3            salary_std            1.165278
9   years_since_joining            0.188016
6          joining_year            0.168968
4         salary_zscore            0.108849
14      avg_word_length            0.073016
0                   age            0.070635
1                salary            0.061071
5        gender_encoded            0.000000
8           joining_day            0.000000
7         joining_month            0.000000
10          salary_lag1            0.000000
11          salary_diff            0.000000
12          text_length            0.000000
13           word_count            0.000000
```

```
# top 5 features using Mutual Information
selector = SelectKBest(score_func=mutual_info_classif, k=5)
X_new = selector.fit_transform(X.fillna(0), y)

selected_features = X.columns[selector.get_support()]
print("\nTop 5 Selected Features:", selected_features.tolist())
```