In [5]:
```
import numpy as np
import pandas as pd
```

In [7]:
```
cancer_df.columns
```

Out[7]:
```
Index(['Patient_ID', 'Age', 'Gender', 'Country_Region', 'Year', 'Genetic_Ri
sk',
       'Air_Pollution', 'Alcohol_Use', 'Smoking', 'Obesity_Level',
       'Cancer_Type', 'Cancer_Stage', 'Treatment_Cost_USD', 'Survival_Year
s',
       'Target_Severity_Score'],
      dtype='object')
```

In [8]:
```
bankrupt_df.columns
```

Out[8]:
```
Index(['Bankrupt?', ' ROA(C) before interest and depreciation before intere
st',
       ' ROA(A) before interest and % after tax',
       ' ROA(B) before interest and depreciation after tax',
       ' Operating Gross Margin', ' Realized Sales Gross Margin',
       ' Operating Profit Rate', ' Pre-tax net Interest Rate',
       ' After-tax net Interest Rate',
       ' Non-industry income and expenditure/revenue',
       ' Continuous interest rate (after tax)', ' Operating Expense Rate',
       ' Research and development expense rate', ' Cash flow rate',
       ' Interest-bearing debt interest rate', ' Tax rate (A)',
       ' Net Value Per Share (B)', ' Net Value Per Share (A)',
       ' Net Value Per Share (C)', ' Persistent EPS in the Last Four Season
s',
       ' Cash Flow Per Share', ' Revenue Per Share (Yuan ¥)',
       ' Operating Profit Per Share (Yuan ¥)',
       ' Per Share Net profit before tax (Yuan ¥)',
       ' Realized Sales Gross Profit Growth Rate',
       ' Operating Profit Growth Rate', ' After-tax Net Profit Growth Rat
e',
       ' Regular Net Profit Growth Rate', ' Continuous Net Profit Growth Ra
te',
       ' Total Asset Growth Rate', ' Net Value Growth Rate',
       ' Total Asset Return Growth Rate Ratio', ' Cash Reinvestment %',
       ' Current Ratio', ' Quick Ratio', ' Interest Expense Ratio',
       ' Total debt/Total net worth', ' Debt ratio %', ' Net worth/Assets',
       ' Long-term fund suitability ratio (A)', ' Borrowing dependency',
       ' Contingent liabilities/Net worth',
       ' Operating profit/Paid-in capital',
       ' Net profit before tax/Paid-in capital',
       ' Inventory and accounts receivable/Net value', ' Total Asset Turnov
er',
       ' Accounts Receivable Turnover', ' Average Collection Days',
       ' Inventory Turnover Rate (times)', ' Fixed Assets Turnover Frequenc
y',
       ' Net Worth Turnover Rate (times)', ' Revenue per person',
       ' Operating profit per person', ' Allocation rate per person',
       ' Working Capital to Total Assets', ' Quick Assets/Total Assets',
       ' Current Assets/Total Assets', ' Cash/Total Assets',
       ' Quick Assets/Current Liability', ' Cash/Current Liability',
       ' Current Liability to Assets', ' Operating Funds to Liability',
       ' Inventory/Working Capital', ' Inventory/Current Liability',
       ' Current Liabilities/Liability', ' Working Capital/Equity',
       ' Current Liabilities/Equity', ' Long-term Liability to Current Asse
ts',
       ' Retained Earnings to Total Assets', ' Total income/Total expense',
       ' Total expense/Assets', ' Current Asset Turnover Rate',
       ' Quick Asset Turnover Rate', ' Working capitcal Turnover Rate',
       ' Cash Turnover Rate', ' Cash Flow to Sales', ' Fixed Assets to Asse
ts',
       ' Current Liability to Liability', ' Current Liability to Equity',
       ' Equity to Long-term Liability', ' Cash Flow to Total Assets',
       ' Cash Flow to Liability', ' CFO to Assets', ' Cash Flow to Equity',
       ' Current Liability to Current Assets', ' Liability-Assets Flag',
       ' Net Income to Total Assets', ' Total assets to GNP price',
       ' No-credit Interval', ' Gross Profit to Sales',
```

```
       ' Net Income to Stockholder's Equity', ' Liability to Equity',
       ' Degree of Financial Leverage (DFL)',
       ' Interest Coverage Ratio (Interest expense to EBIT)',
       ' Net Income Flag', ' Equity to Liability'],
      dtype='object')
```

In [7]:
```
# CS7641 Supervised Learning Report — Jupyter Notebook Version
import os, json, time
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.model_selection import StratifiedKFold, learning_curve, cross_v
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier

RANDOM_STATE = 42
OUTDIR = "outputs"
os.makedirs(OUTDIR, exist_ok=True)
```

In [10]:
```
def bin_severity(y_cont: pd.Series, bins: int = 3) -> pd.Series:
    return pd.qcut(y_cont, q=bins, labels=False, duplicates='drop')

def preprocess_cancer(df: pd.DataFrame, target_bins: int = 3):
    y = bin_severity(df["Target_Severity_Score"], bins=target_bins).astype(i
    X = df.drop(columns=["Target_Severity_Score", "Patient_ID"], errors="igr

    cat_cols = X.select_dtypes(include=["object", "category"]).columns.tolis
    num_cols = X.select_dtypes(include=[np.number]).columns.tolist()

    numeric_tf = Pipeline([
        ("imputer", SimpleImputer(strategy="median")),
        ("scaler", StandardScaler())
    ])
    categorical_tf = Pipeline([
        ("imputer", SimpleImputer(strategy="most_frequent")),
        ("onehot", OneHotEncoder(handle_unknown="ignore"))
    ])

    pre = ColumnTransformer([
        ("num", numeric_tf, num_cols),
        ("cat", categorical_tf, cat_cols)
    ])
    return X, y, pre

def preprocess_bankruptcy(df: pd.DataFrame):
    df.columns = [c.strip() for c in df.columns]
    y = df["Bankrupt?"].astype(int).values
    X = df.drop(columns=["Bankrupt?"], errors="ignore")
```

```
    num_cols = X.columns.tolist()
    pre = ColumnTransformer([
        ("num", Pipeline([
            ("imputer", SimpleImputer(strategy="median")),
            ("scaler", StandardScaler())
        ]), num_cols)
    ])
    return X, y, pre

def metric_dict(y_true, y_pred, y_proba=None, is_binary=False):
    d = {
        "accuracy": float(accuracy_score(y_true, y_pred)),
        "f1_macro": float(f1_score(y_true, y_pred, average="macro"))
    }
    if is_binary and y_proba is not None:
        try:
            d["roc_auc"] = float(roc_auc_score(y_true, y_proba))
        except Exception:
            d["roc_auc"] = None
    else:
        d["roc_auc"] = None
    return d

def plot_learning_curve(estimator, X, y, pre, title, filename, cv_splits=5):
    pipe = Pipeline([("pre", pre), ("clf", estimator)])
    cv = StratifiedKFold(n_splits=cv_splits, shuffle=True, random_state=RAND
    train_sizes, train_scores, test_scores, fit_times, _ = learning_curve(
        pipe, X, y, cv=cv, scoring="f1_macro", n_jobs=-1,
        train_sizes=np.linspace(0.1, 1.0, 5), return_times=True
    )

    train_mean = np.mean(train_scores, axis=1)
    test_mean = np.mean(test_scores, axis=1)

    plt.figure()
    plt.title(title)
    plt.xlabel("Training examples")
    plt.ylabel("F1 Macro")
    plt.plot(train_sizes, train_mean, marker="o", label="Training score")
    plt.plot(train_sizes, test_mean, marker="o", label="Cross-validation sco
    plt.legend()
    plt.grid(True, linestyle=":")
    plt.tight_layout()
    plt.savefig(os.path.join(OUTDIR, filename), dpi=200)
    plt.show()
```

In [11]:

```
def crossval_metrics(pipe, X, y, cv_splits=5):
    cv = StratifiedKFold(n_splits=cv_splits, shuffle=True, random_state=RAND
    y_pred = cross_val_predict(pipe, X, y, cv=cv, n_jobs=-1, method="predict
    is_binary = len(np.unique(y)) == 2
    y_proba = None
    try:
        if is_binary:
            y_proba = cross_val_predict(pipe, X, y, cv=cv, n_jobs=-1, methoc
    except Exception:
        pass
```

```
    return metric_dict(y, y_pred, y_proba=y_proba, is_binary=is_binary)

def sweep_and_plot(param_grid, pipe, X, y, title_prefix, out_prefix, cv_spli
    (param_name, values), = param_grid.items()
    scores = []
    for v in values:
        pipe.set_params(**{f"clf__{param_name}": v})
        metrics = crossval_metrics(pipe, X, y, cv_splits=cv_splits)
        scores.append((v, metrics["f1_macro"]))

    xs, ys = zip(*scores)
    plt.figure()
    plt.title(f"{title_prefix}: {param_name}")
    plt.xlabel(param_name)
    plt.ylabel("F1 Macro (CV)")
    plt.plot(xs, ys, marker="o")
    plt.grid(True, linestyle=":")
    plt.tight_layout()
    plt.savefig(os.path.join(OUTDIR, f"{out_prefix}__{param_name}.png"), dpi
    plt.show()
```

In [12]:

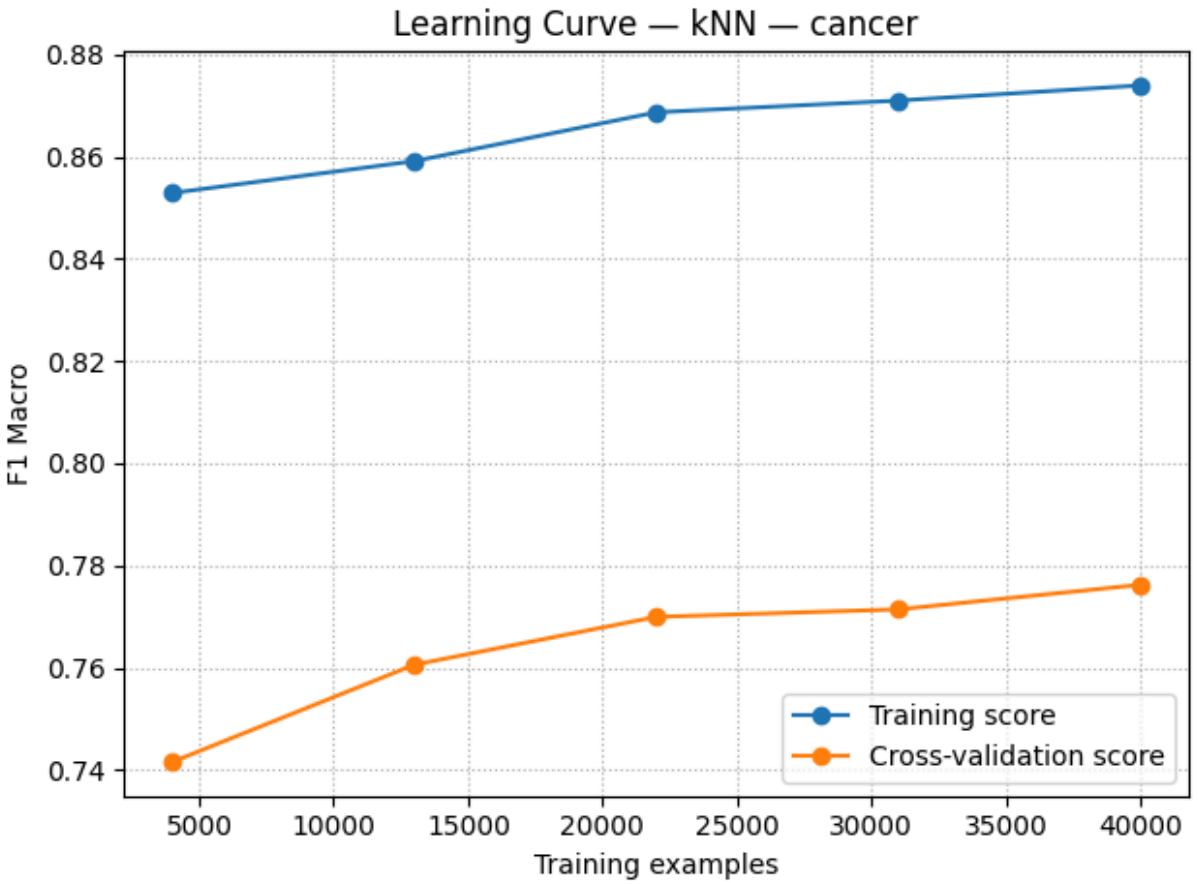
```
def run_for_dataset(dataset, X, y, pre):
    results = {}

    # --- kNN ---
    knn = KNeighborsClassifier(n_neighbors=5)
    plot_learning_curve(knn, X, y, pre, f"Learning Curve — kNN — {dataset}",
    pipe_knn = Pipeline([("pre", pre), ("clf", KNeighborsClassifier())])
    sweep_and_plot({"n_neighbors": [1,3,5,7,11,15,21]}, pipe_knn, X, y, f"Mc
    results["knn"] = crossval_metrics(Pipeline([("pre", pre), ("clf", KNeigh

    # --- SVM RBF ---
    pipe_svm_rbf = Pipeline([("pre", pre), ("clf", SVC(kernel="rbf", class_w
    plot_learning_curve(pipe_svm_rbf.named_steps["clf"], X, y, pre, f"Learni
    sweep_and_plot({"C": [1e-2, 1e-1, 1, 10, 100]}, pipe_svm_rbf, X, y, f"SV
    results["svm_rbf"] = crossval_metrics(pipe_svm_rbf, X, y)

    # --- SVM Poly ---
    pipe_svm_poly = Pipeline([("pre", pre), ("clf", SVC(kernel="poly", degre
    plot_learning_curve(pipe_svm_poly.named_steps["clf"], X, y, pre, f"Learr
    sweep_and_plot({"degree": [2,3,4]}, pipe_svm_poly, X, y, f"SVM Poly — {d
    results["svm_poly"] = crossval_metrics(pipe_svm_poly, X, y)

    # --- MLP ReLU / tanh ---
    for act in ["relu", "tanh"]:
        mlp = MLPClassifier(activation=act, hidden_layer_sizes=(128,), alpha
        plot_learning_curve(mlp, X, y, pre, f"Learning Curve — MLP ({act}) -
        pipe_mlp = Pipeline([("pre", pre), ("clf", mlp)])
        sweep_and_plot({"alpha": [1e-5, 1e-4, 1e-3, 1e-2]}, pipe_mlp, X, y,
        results[f"mlp_{act}"] = crossval_metrics(pipe_mlp, X, y)

    return results
```

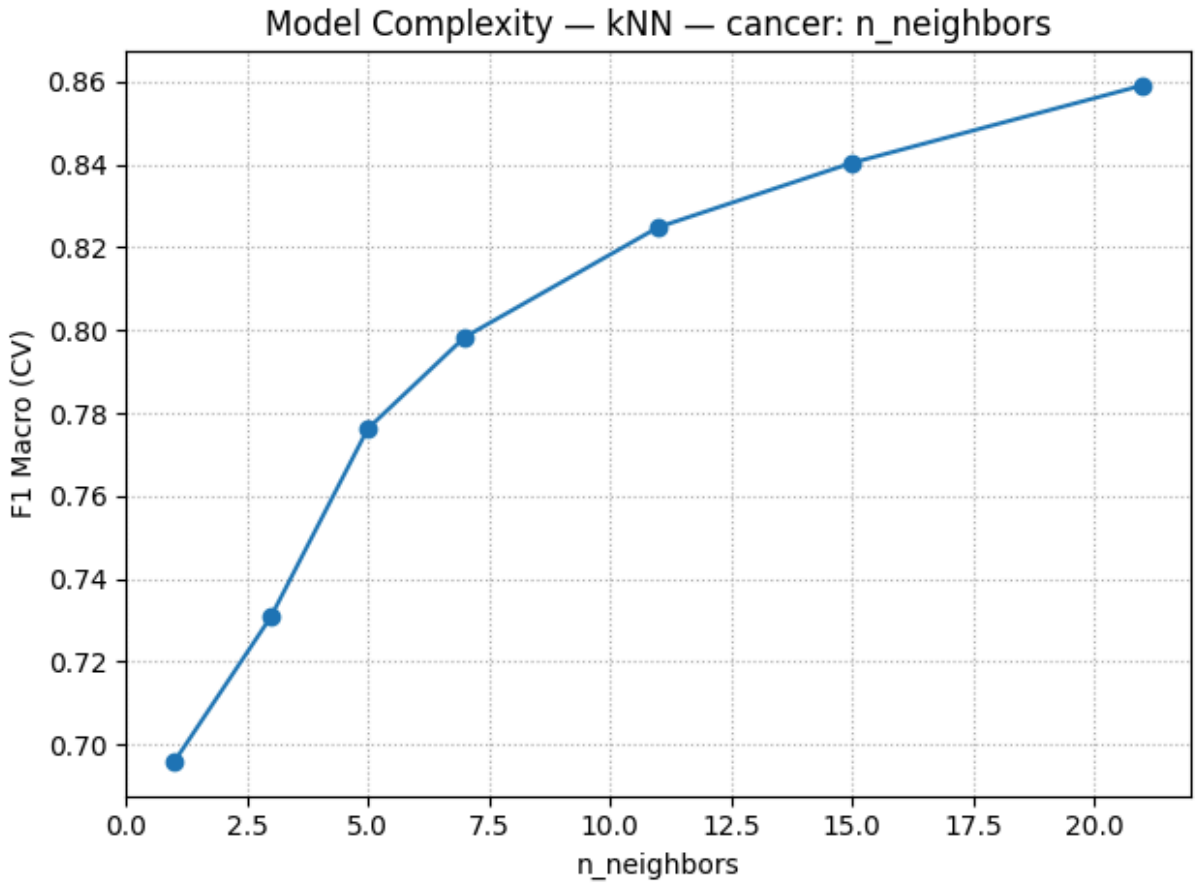
In [14]:

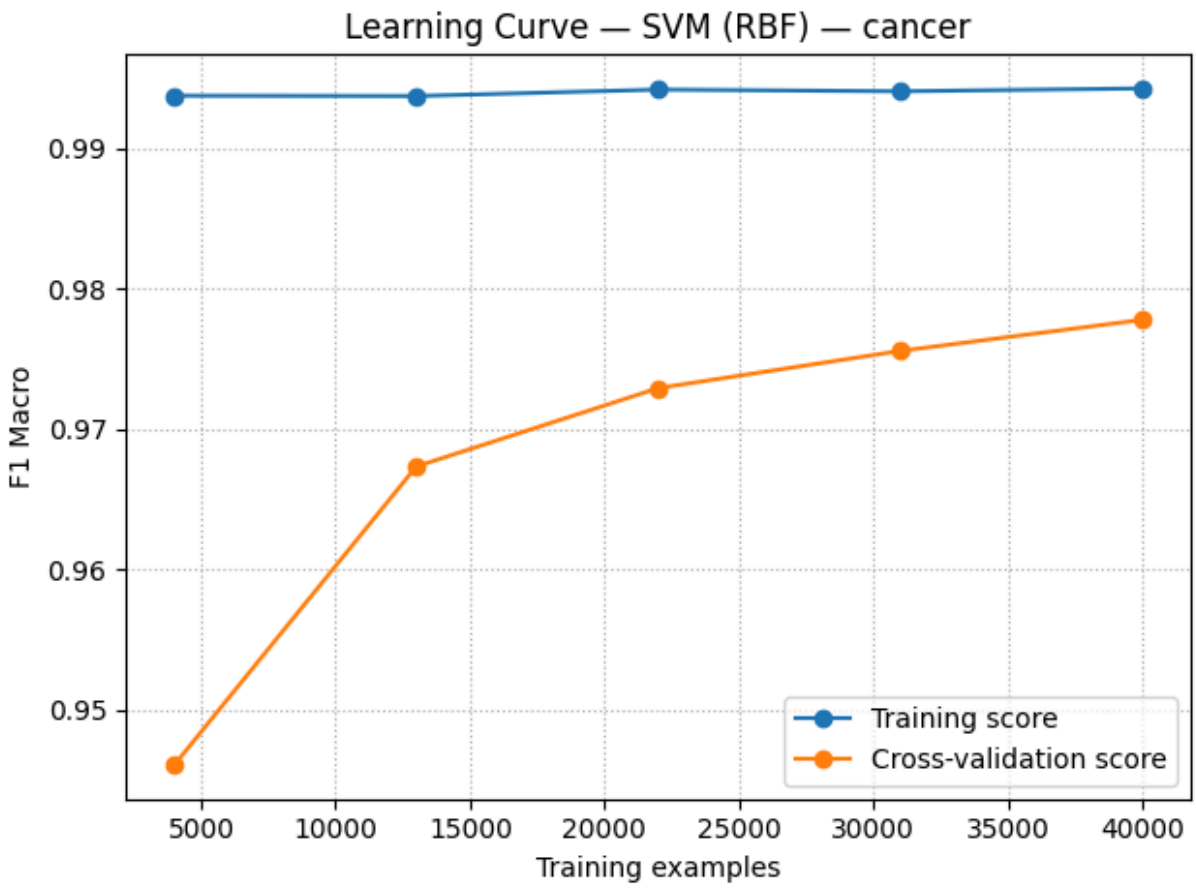
```
cancer_df = pd.read_csv('cancer.csv')
bankrupt_df = pd.read_csv('bankruptcy.csv')
```

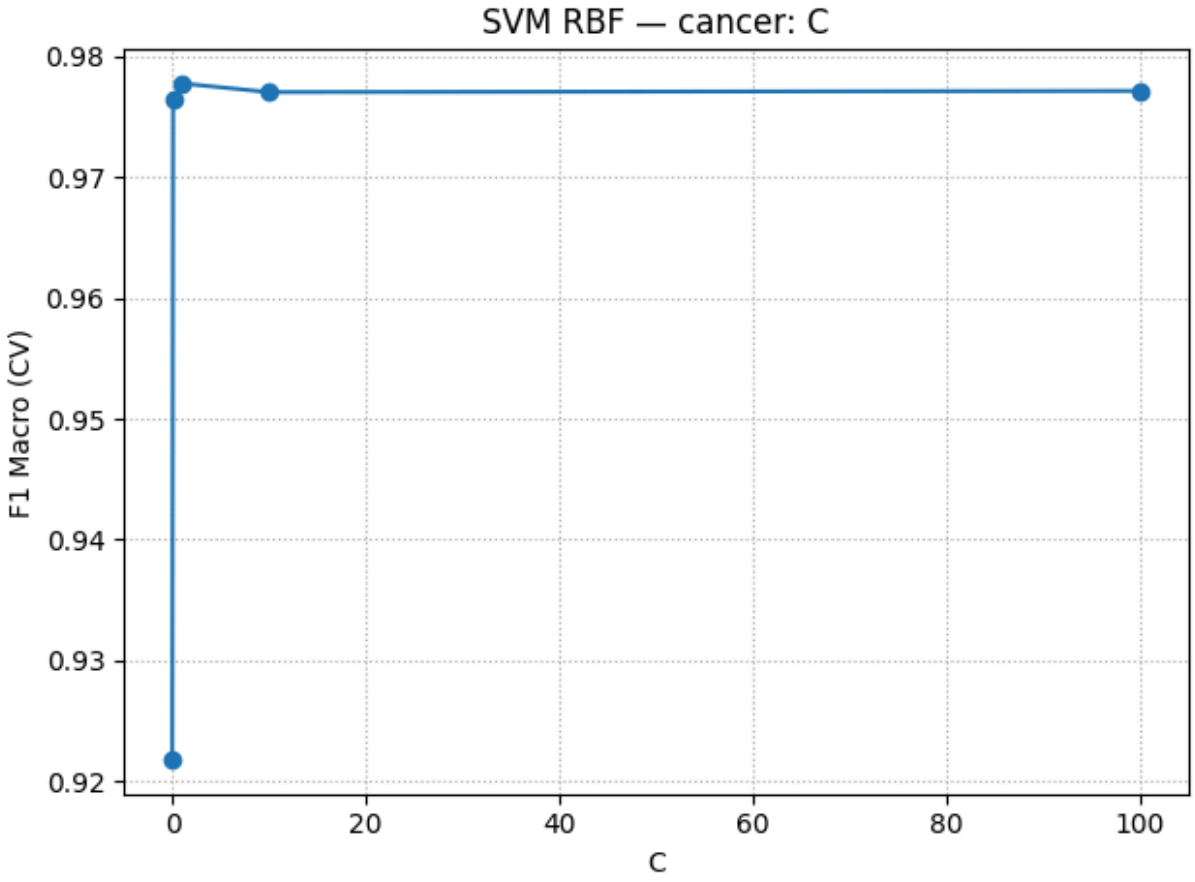
```
Xc, yc, pre_c = preprocess_cancer(cancer_df)
Xb, yb, pre_b = preprocess_bankruptcy(bankrupt_df)

print("Running Cancer Experiments...")
metrics_cancer = run_for_dataset("cancer", Xc, yc, pre_c)

print("Running Bankruptcy Experiments...")
metrics_bankrupt = run_for_dataset("bankrupt", Xb, yb, pre_b)

# Save metrics
with open(os.path.join(OUTDIR, "metrics_all.json"), "w") as f:
    json.dump({"cancer": metrics_cancer, "bankrupt": metrics_bankrupt}, f, i

metrics_cancer, metrics_bankrupt
```

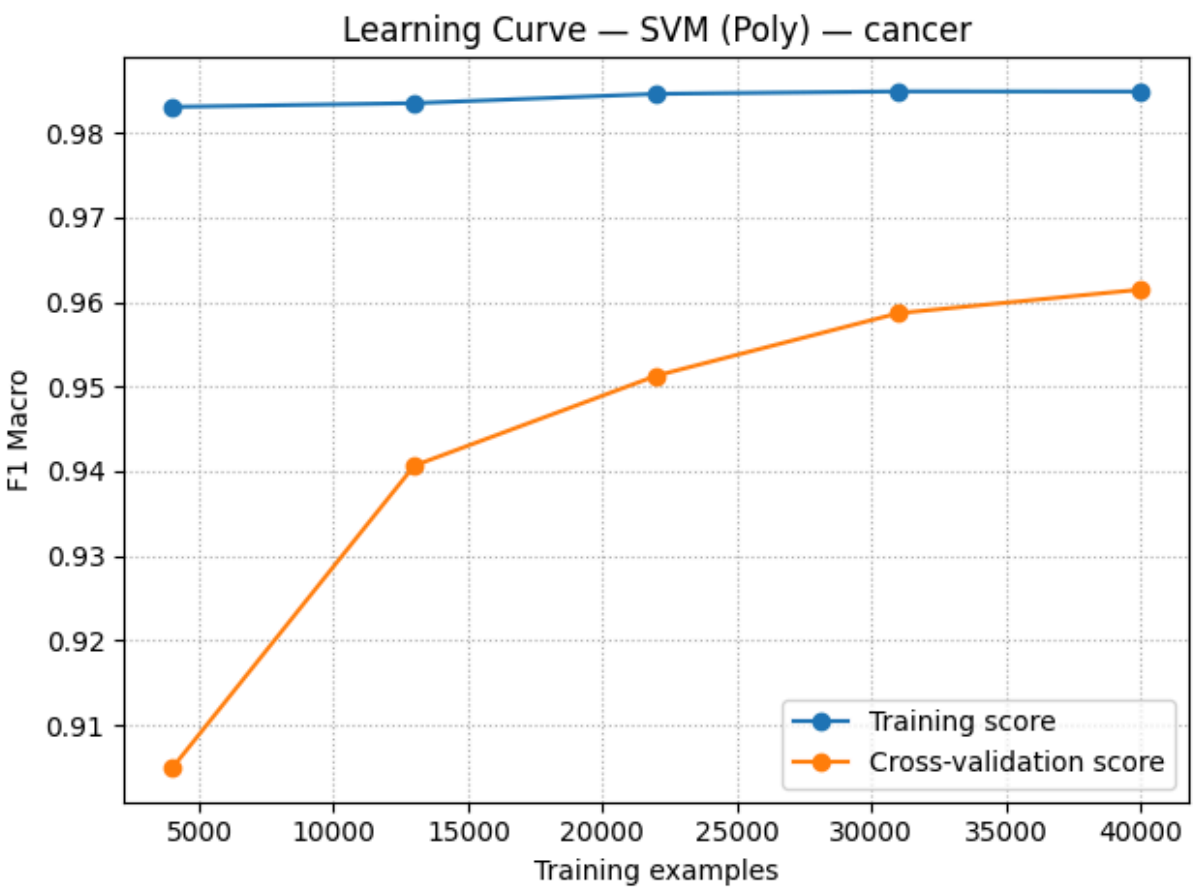
```
Running Cancer Experiments...
```

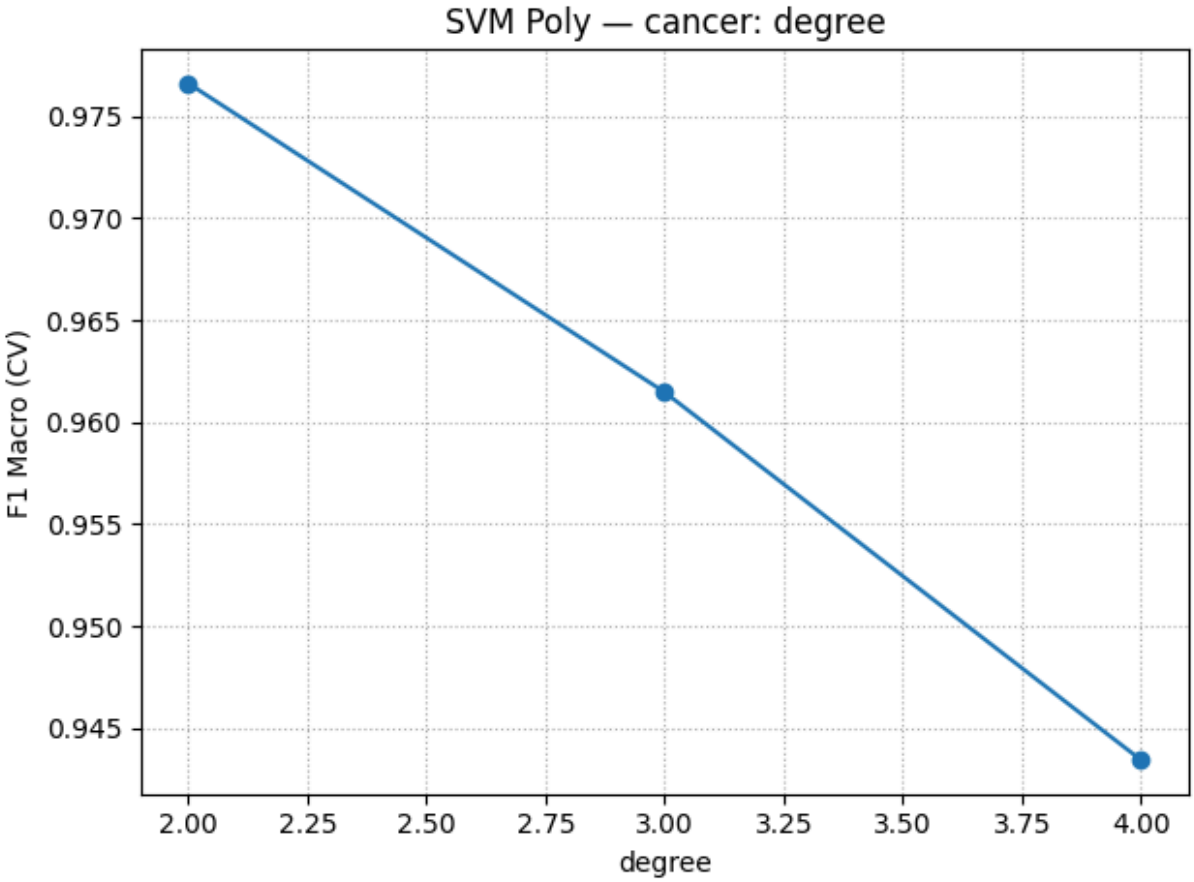


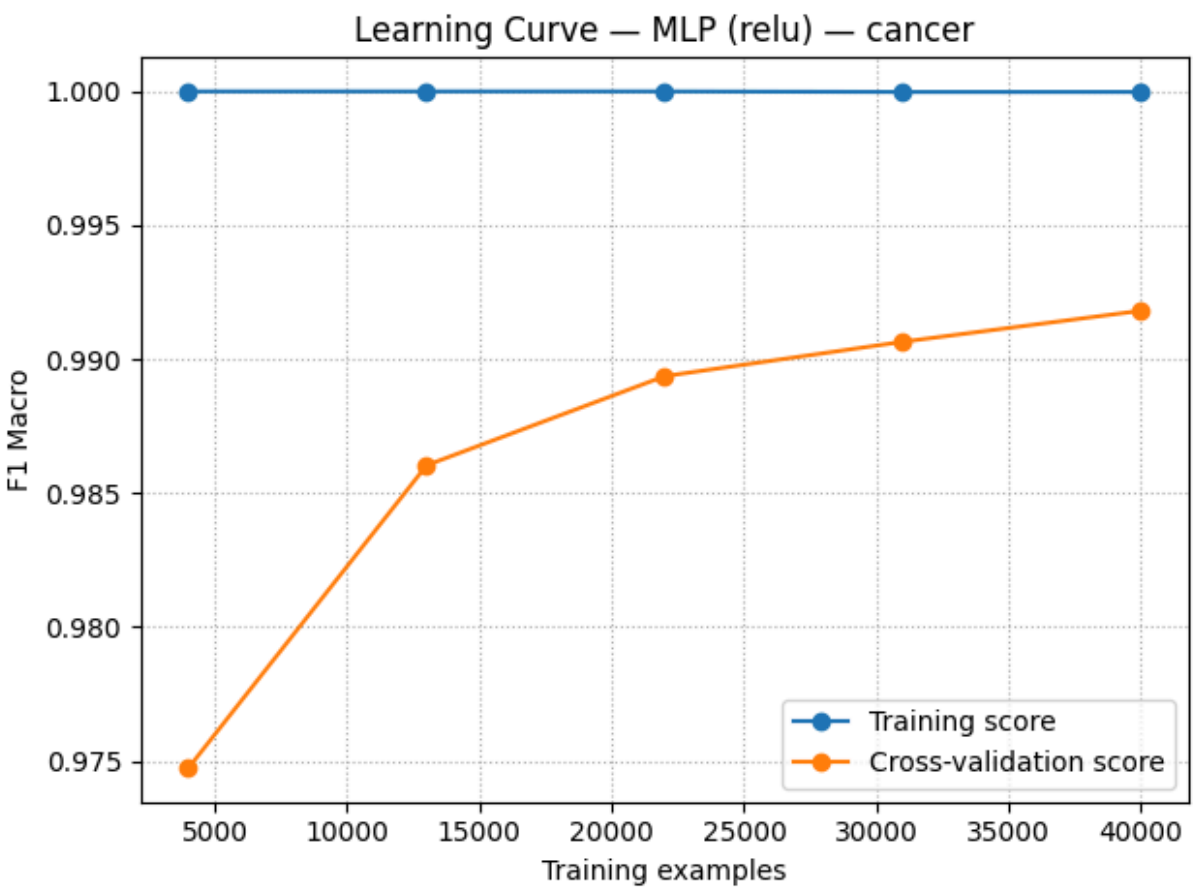

Model Complexity — kNN — cancer: n_neighbors
F1 Macro (CV)
0.86
0.84
0.82
0.80
0.78
0.76
0.74
0.72
0.70
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
n_neighbors

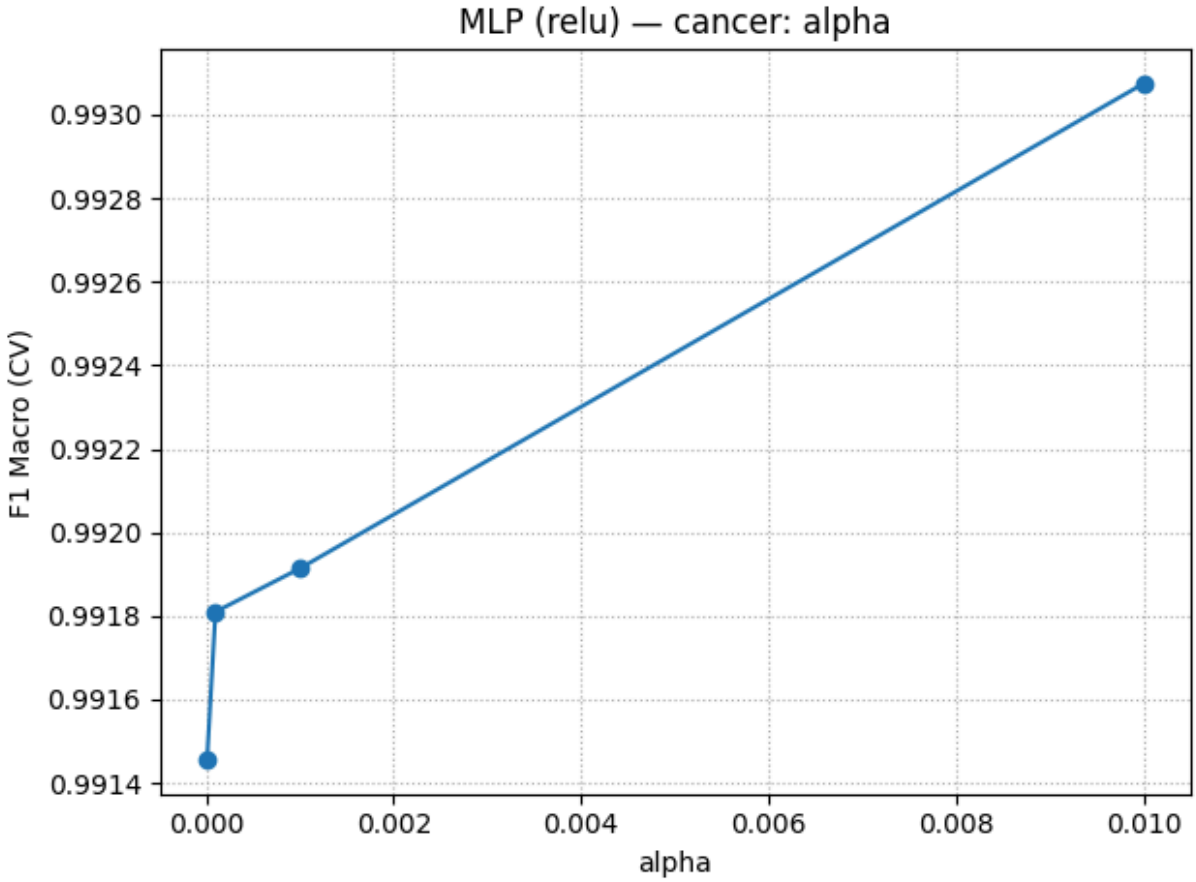

Learning Curve — SVM (RBF) — cancer
F1 Macro
0.99
0.98
0.97
0.96
0.95
5000
10000
15000
20000
25000
30000
35000
40000
Training examples
Training score
Cross-validation score

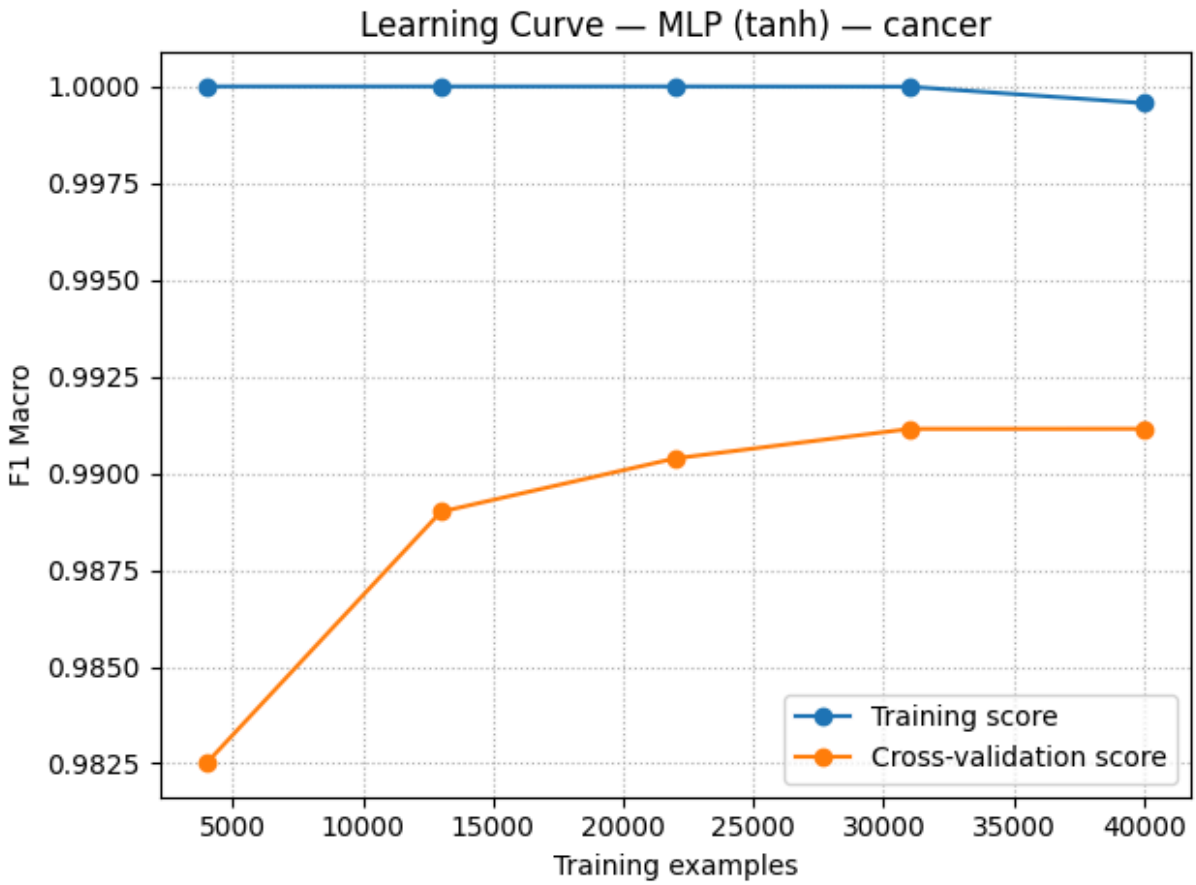

SVM RBF — cancer: C
F1 Macro (CV)
0.98
0.97
0.96
0.95
0.94
0.93
0.92
0
20
40
60
80
100
C

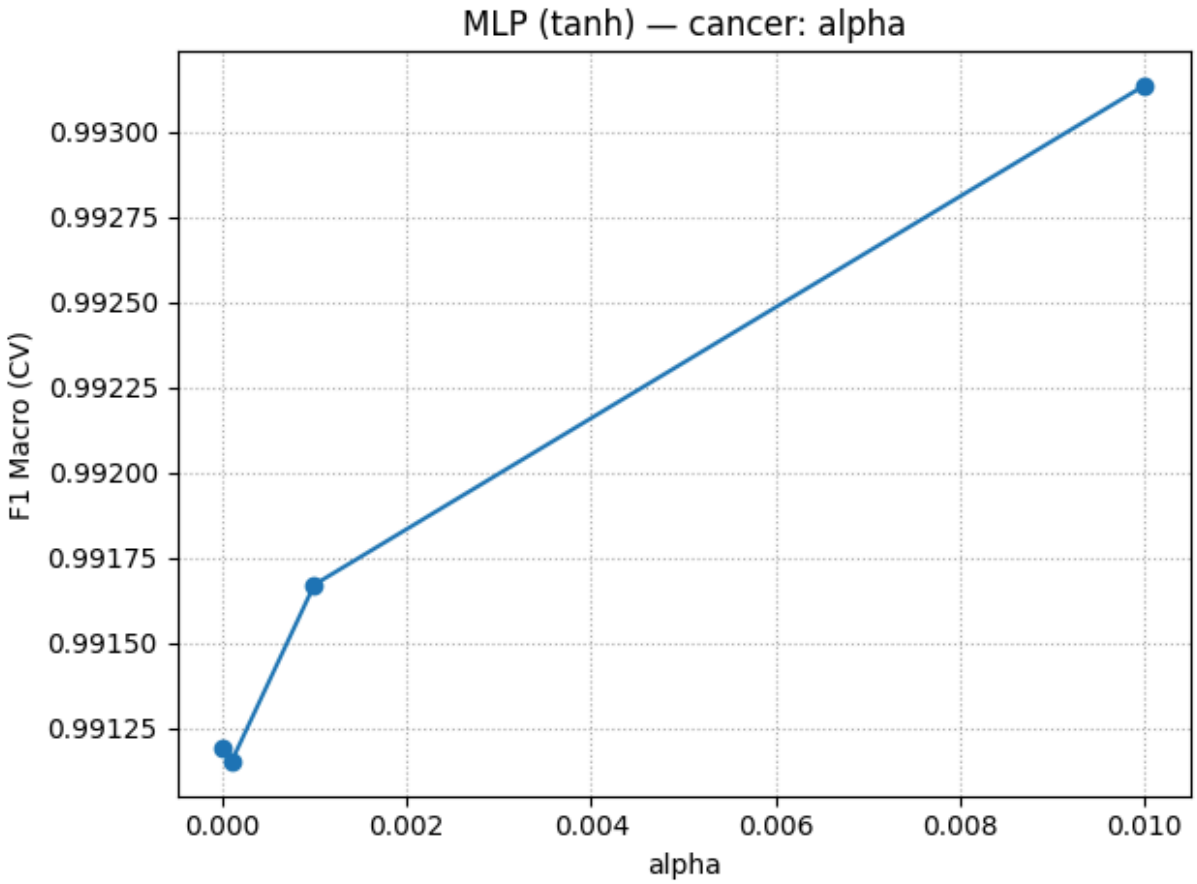

Learning Curve — SVM (Poly) — cancer
F1 Macro
0.98
0.97
0.96
0.95
0.94
0.93
0.92
0.91
5000
10000
15000
20000
25000
30000
35000
40000
Training examples
Training score
Cross-validation score


SVM Poly — cancer: degree
F1 Macro (CV)
0.975
0.970
0.965
0.960
0.955
0.950
0.945
2.00
2.25
2.50
2.75
3.00
3.25
3.50
3.75
4.00
degree

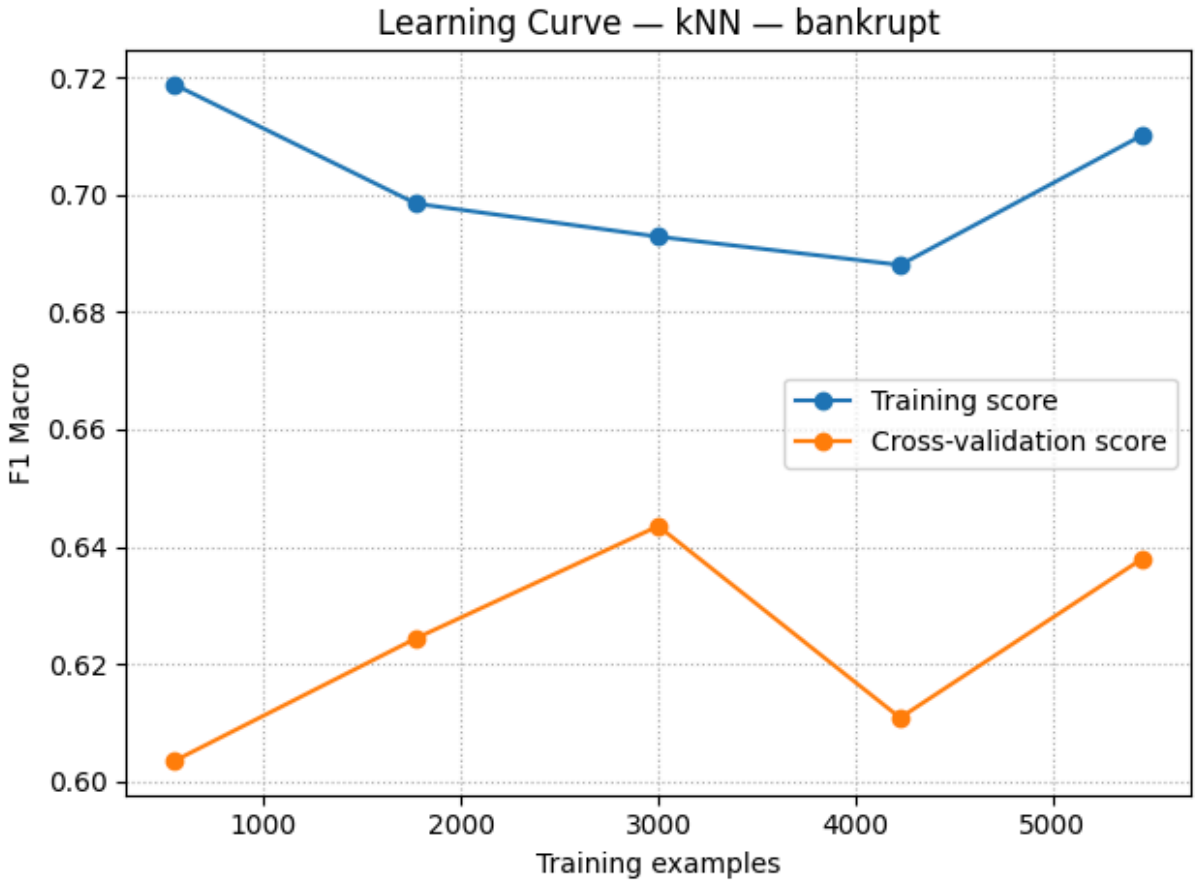

Learning Curve — MLP (relu) — cancer
F1 Macro
1.000
0.995
0.990
0.985
0.980
0.975
Training score
Cross-validation score
5000
10000
15000
20000
25000
30000
35000
40000
Training examples

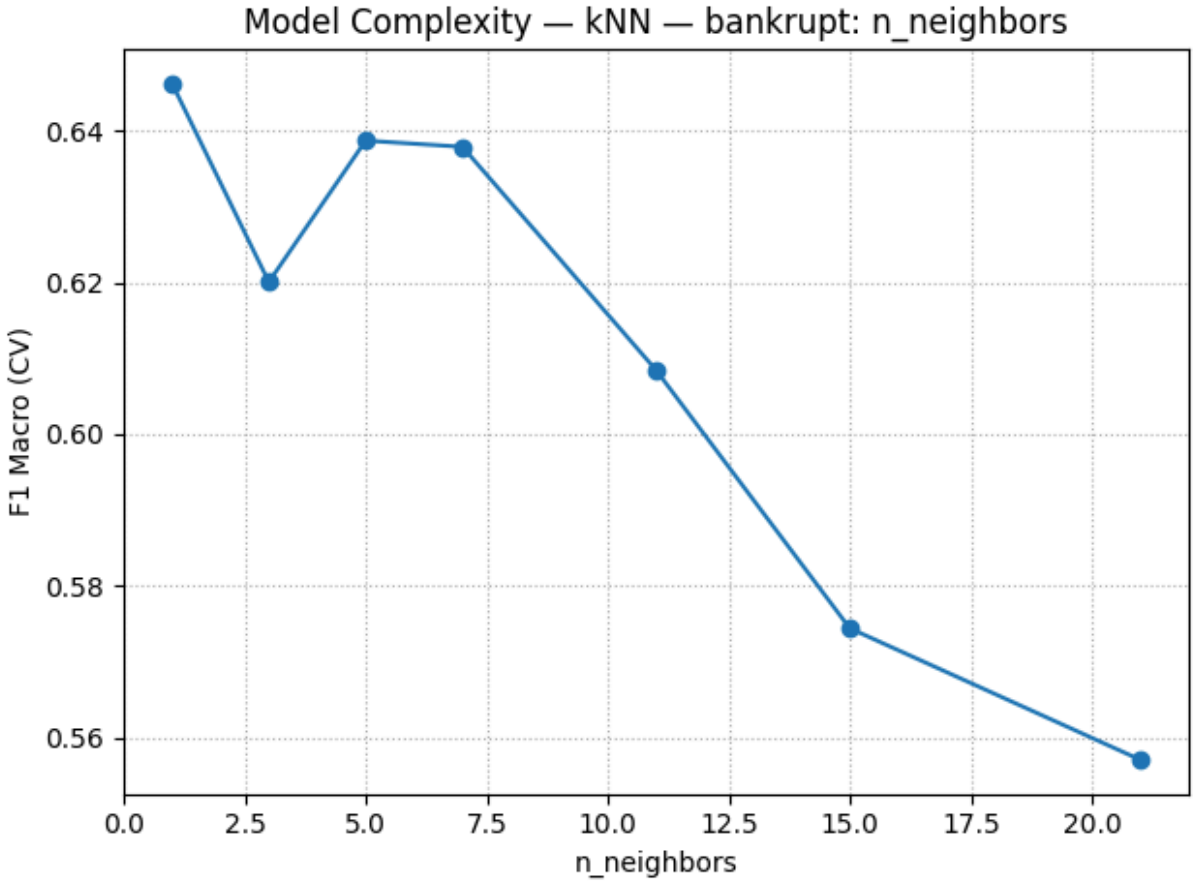

MLP (relu) — cancer: alpha
F1 Macro (CV)
0.9930
0.9928
0.9926
0.9924
0.9922
0.9920
0.9918
0.9916
0.9914
0.000
0.002
0.004
0.006
0.008
0.010
alpha

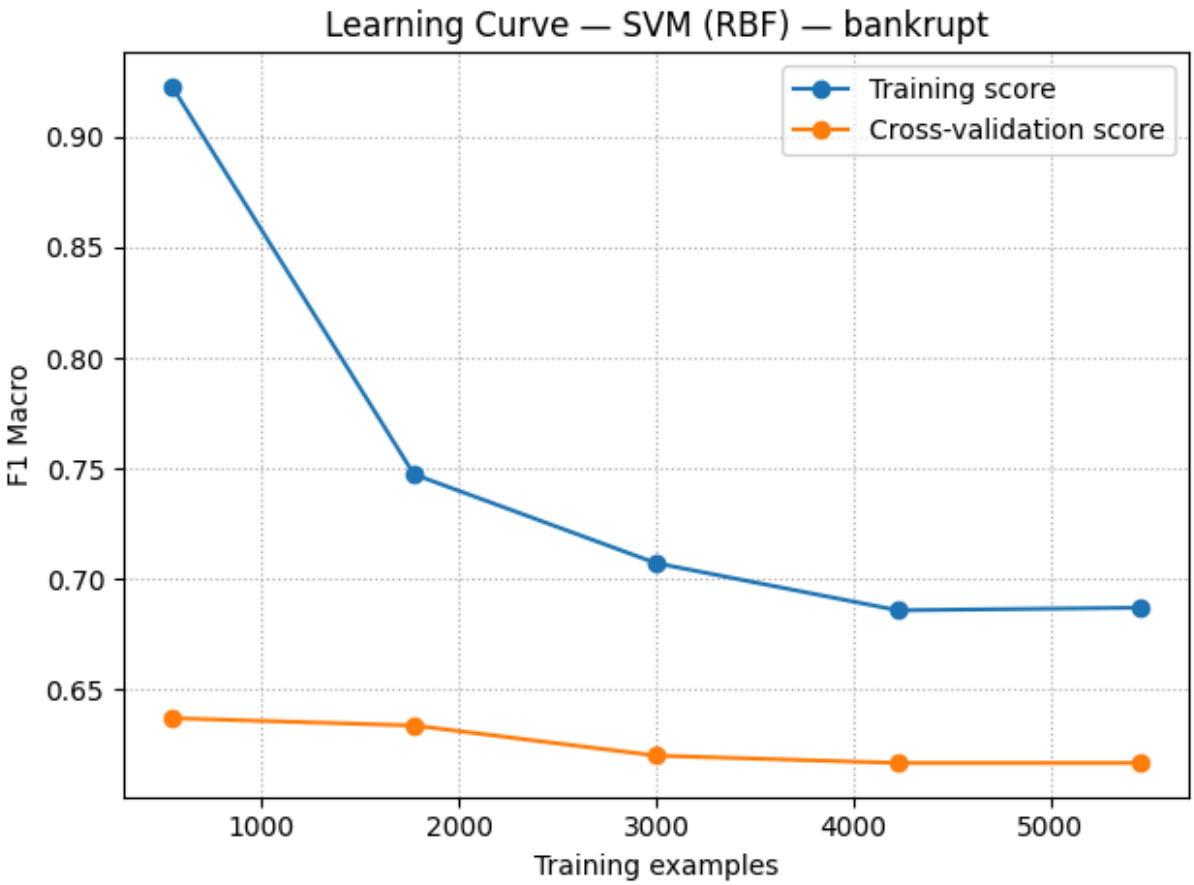

Learning Curve — MLP (tanh) — cancer
F1 Macro
1.0000
0.9975
0.9950
0.9925
0.9900
0.9875
0.9850
0.9825
5000
10000
15000
20000
25000
30000
35000
40000
Training examples
Training score
Cross-validation score

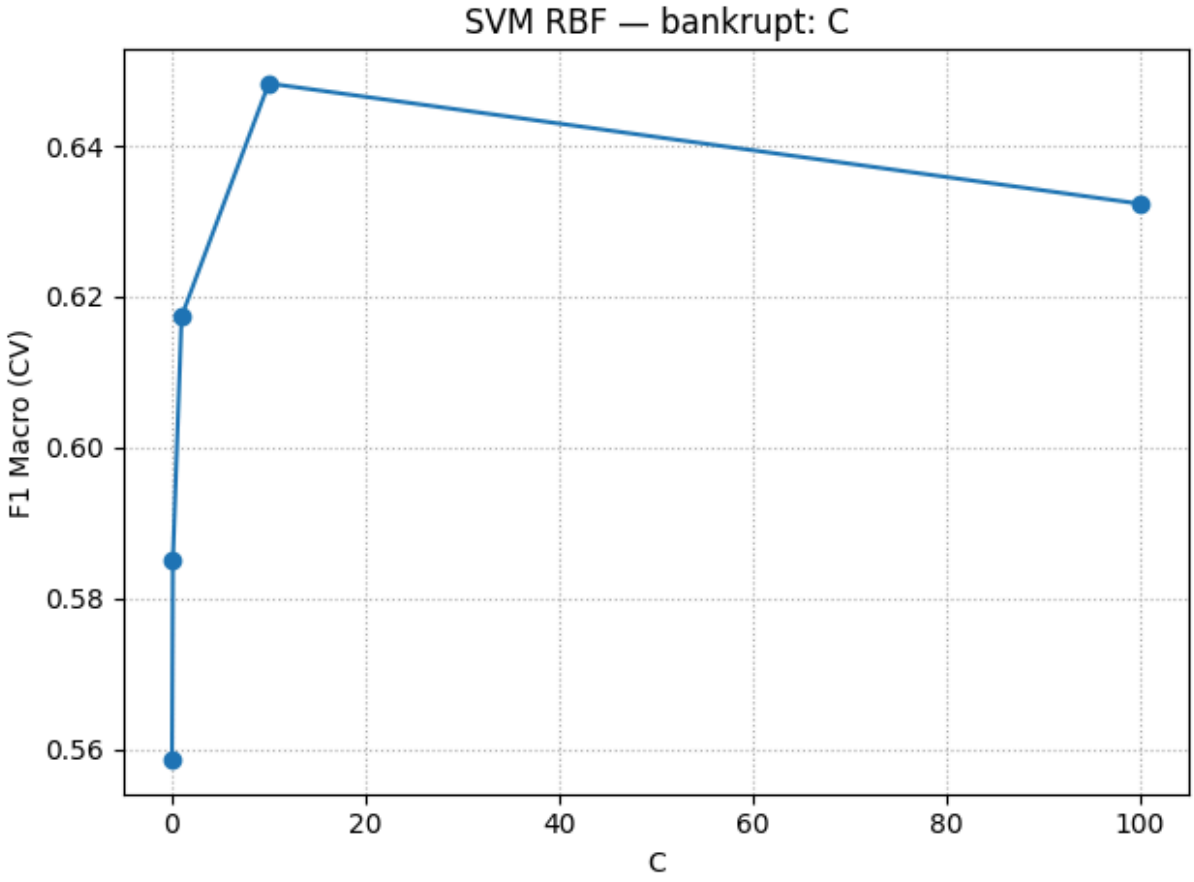


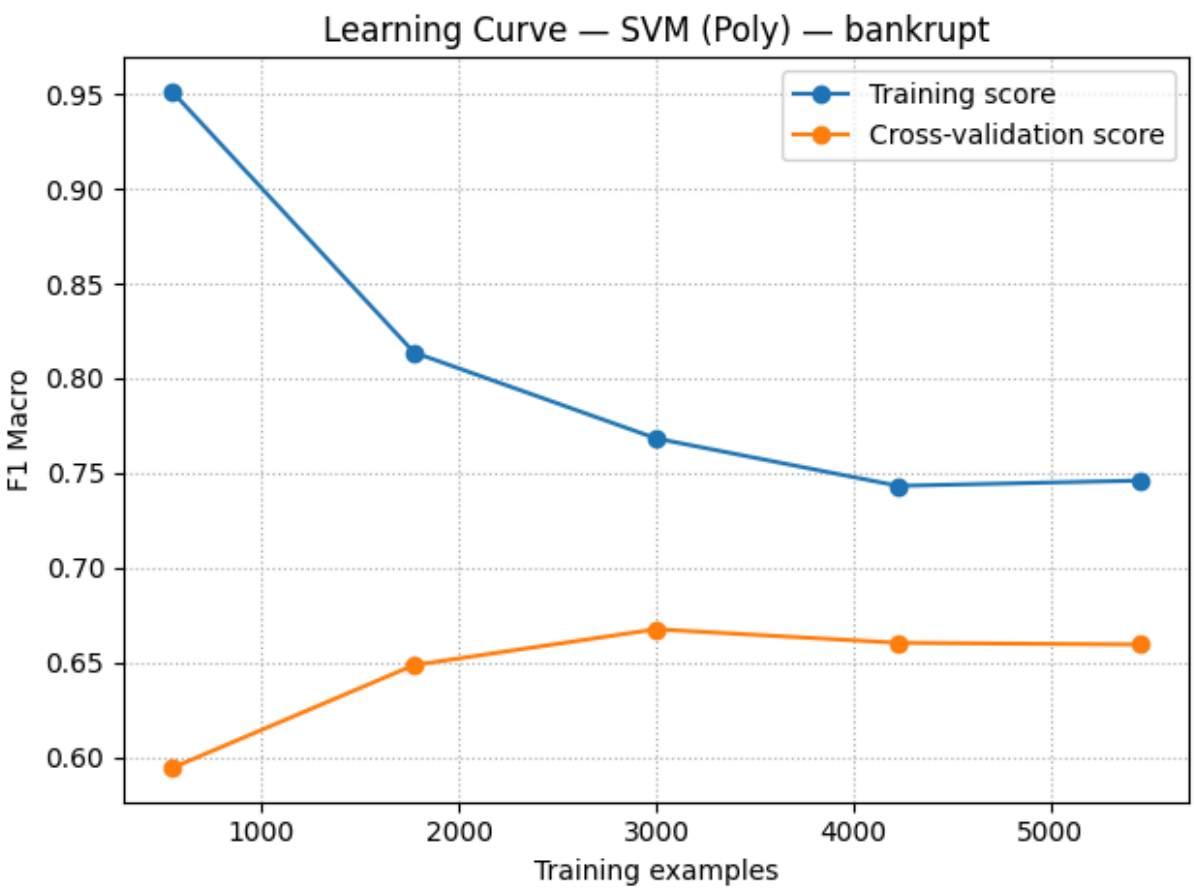

```
Running Bankruptcy Experiments...
```

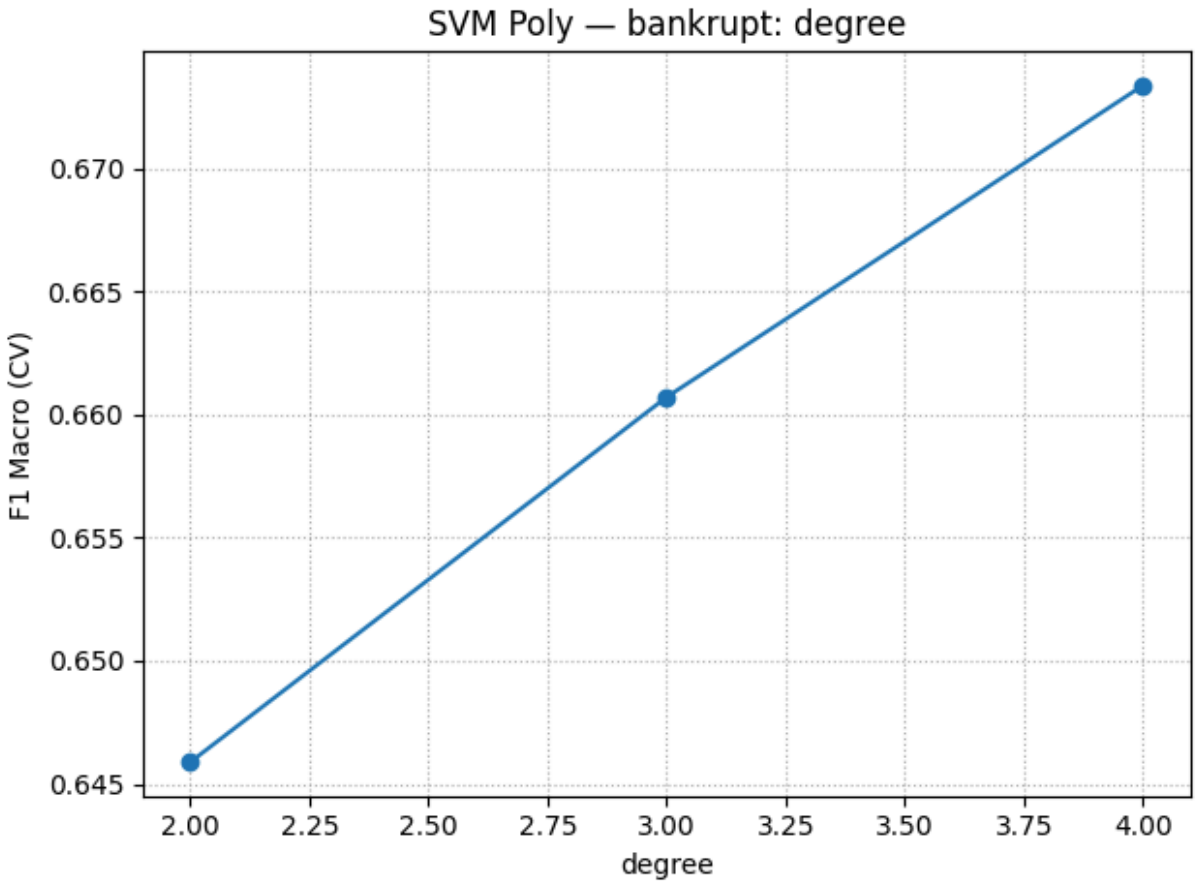


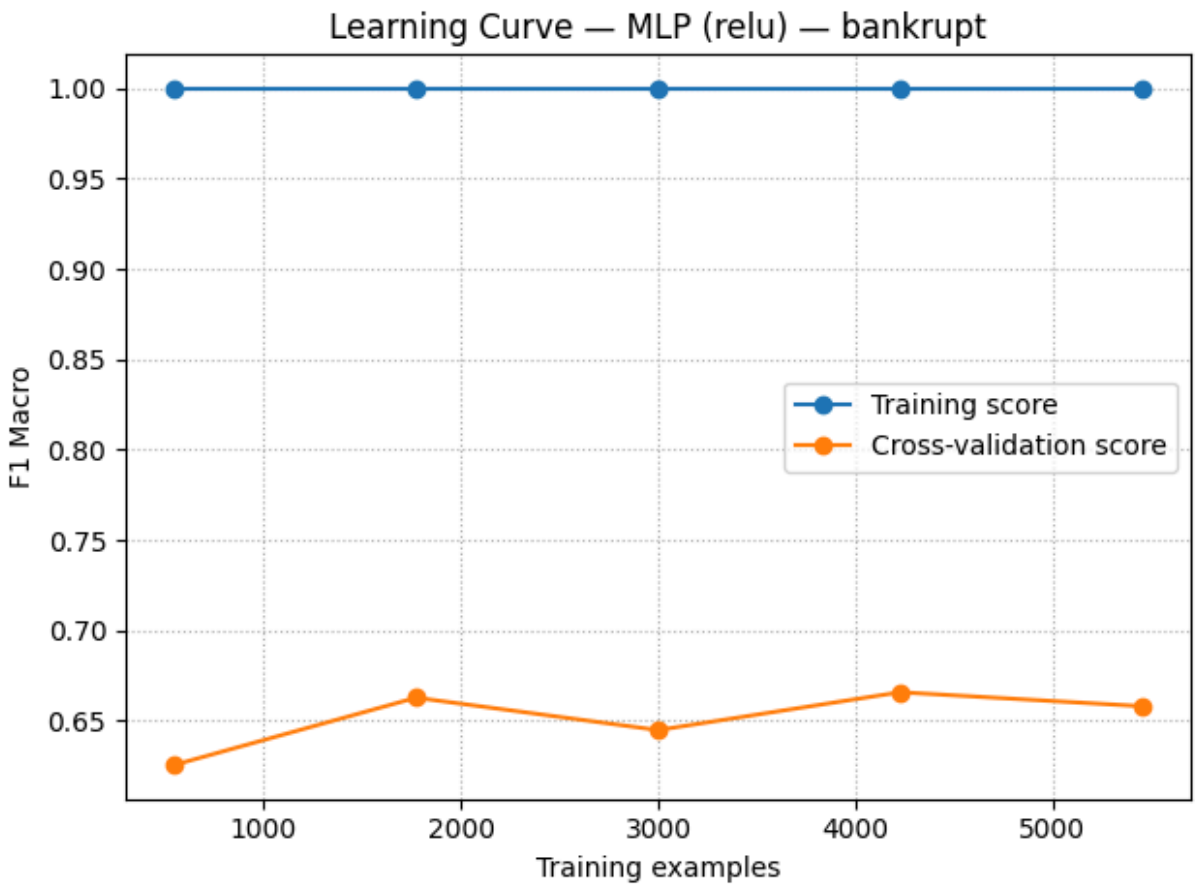

Model Complexity — kNN — bankrupt: n_neighbors
F1 Macro (CV)
0.64
0.62
0.60
0.58
0.56
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
n_neighbors

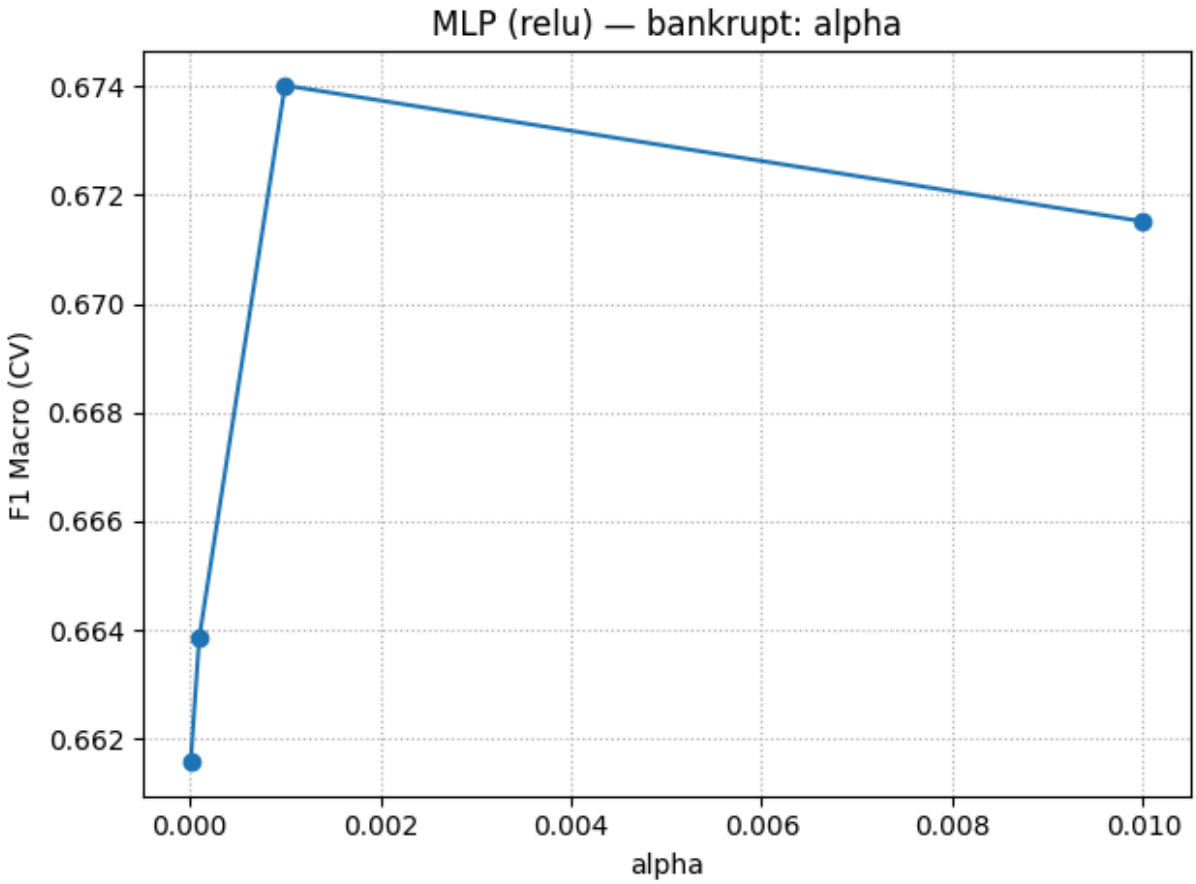

Learning Curve — SVM (RBF) — bankrupt
Training score
Cross-validation score
F1 Macro
0.90
0.85
0.80
0.75
0.70
0.65
1000
2000
3000
4000
5000
Training examples

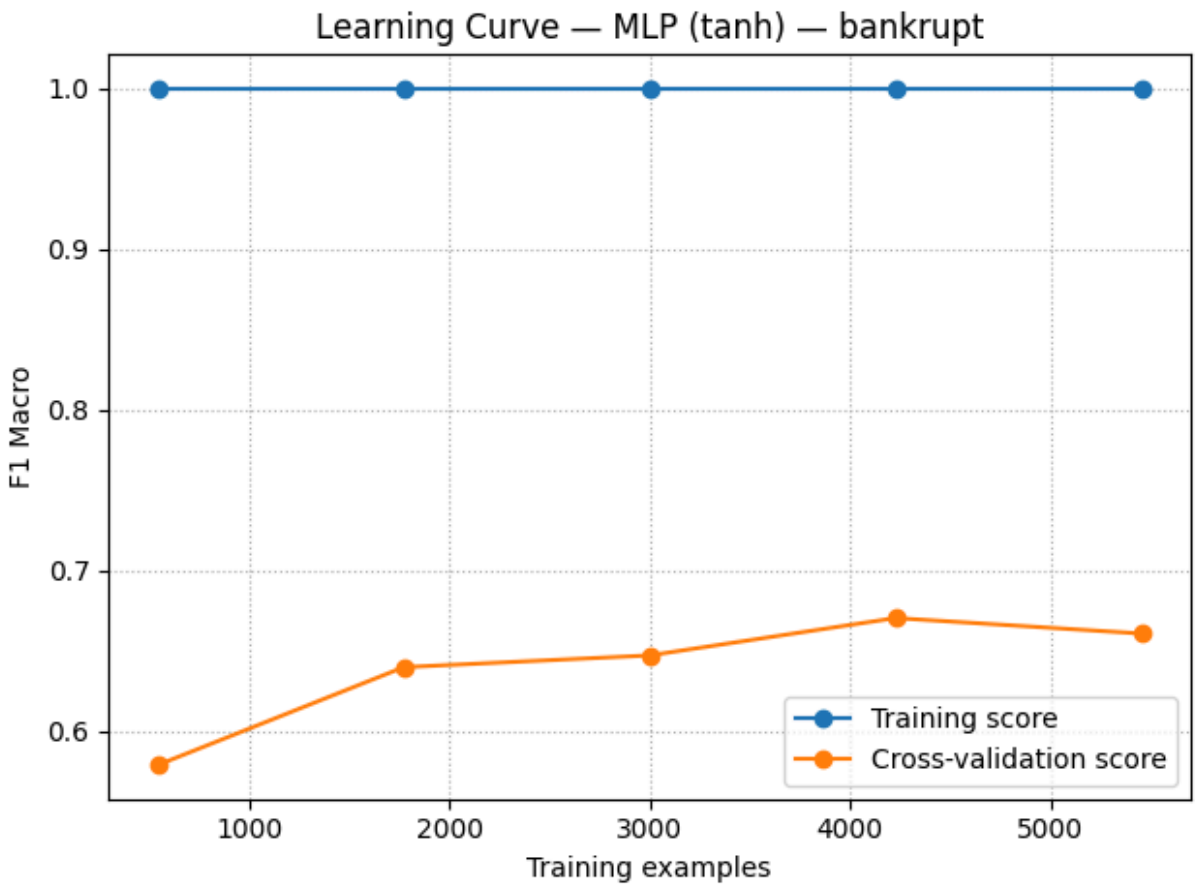

SVM RBF — bankrupt: C
F1 Macro (CV)
0.64
0.62
0.60
0.58
0.56
0
20
40
60
80
100
C

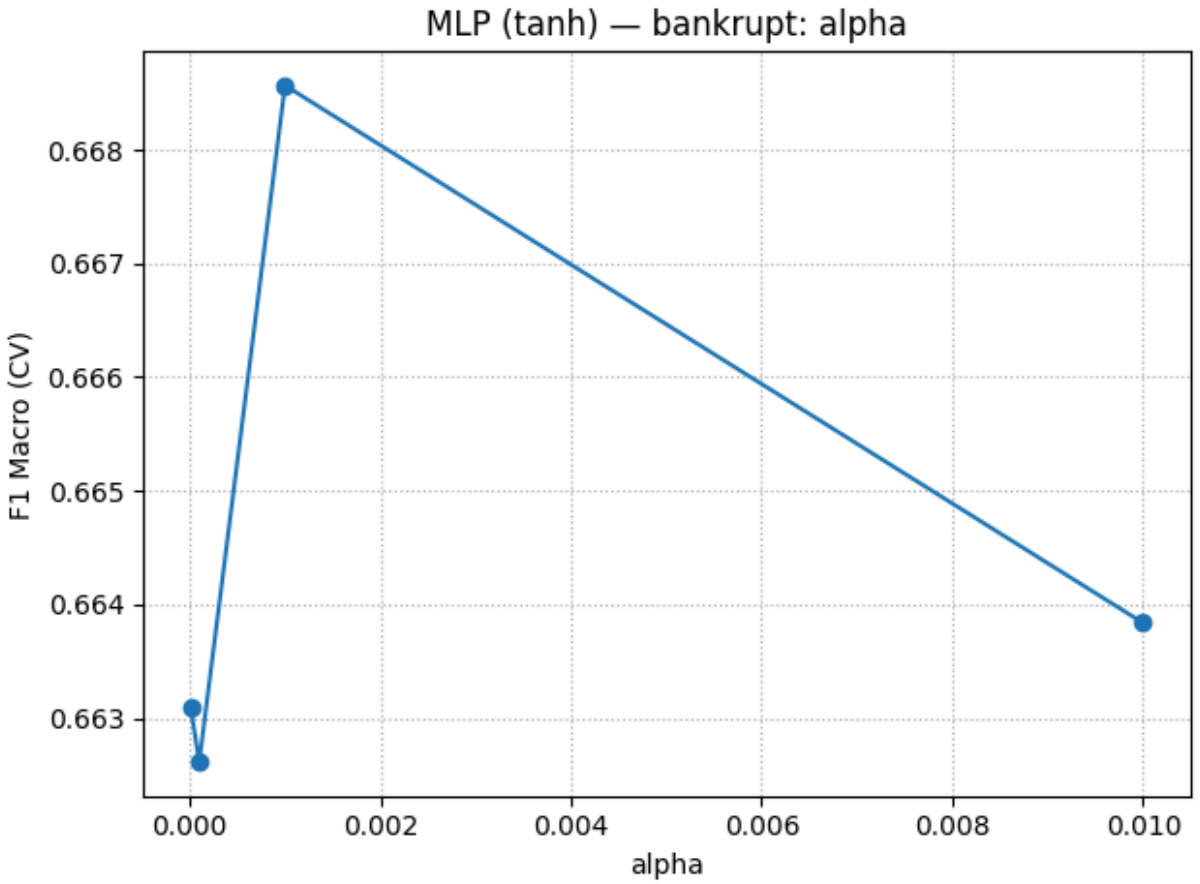

Learning Curve — SVM (Poly) — bankrupt
Training score
Cross-validation score
F1 Macro
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
1000
2000
3000
4000
5000
Training examples


SVM Poly — bankrupt: degree
F1 Macro (CV)
0.670
0.665
0.660
0.655
0.650
0.645
2.00
2.25
2.50
2.75
3.00
3.25
3.50
3.75
4.00
degree


Learning Curve — MLP (relu) — bankrupt
F1 Macro
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
Training score
Cross-validation score
1000
2000
3000
4000
5000
Training examples


MLP (relu) — bankrupt: alpha
F1 Macro (CV)
0.674
0.672
0.670
0.668
0.666
0.664
0.662
0.000
0.002
0.004
0.006
0.008
0.010
alpha


Learning Curve — MLP (tanh) — bankrupt
F1 Macro
1.0
0.9
0.8
0.7
0.6
1000
2000
3000
4000
5000
Training examples
Training score
Cross-validation score



Out[14]:
```
({'knn': {'accuracy': 0.7972, 'f1_macro': 0.7983019048097747, 'roc_auc': No
ne},
  'svm_rbf': {'accuracy': 0.97718,
   'f1_macro': 0.9771597106048683,
   'roc_auc': None},
  'svm_poly': {'accuracy': 0.94298,
   'f1_macro': 0.9434422206645209,
   'roc_auc': None},
  'mlp_relu': {'accuracy': 0.99308,
   'f1_macro': 0.9930748912025749,
   'roc_auc': None},
  'mlp_tanh': {'accuracy': 0.99314,
   'f1_macro': 0.9931369828790962,
   'roc_auc': None}},
 {'knn': {'accuracy': 0.9706701862443173,
   'f1_macro': 0.6379027992837707,
   'roc_auc': 0.8035656228905206},
  'svm_rbf': {'accuracy': 0.9530722979909078,
   'f1_macro': 0.63231220301284,
   'roc_auc': 0.8526973783906652},
  'svm_poly': {'accuracy': 0.9501393166153395,
   'f1_macro': 0.6733658005486591,
   'roc_auc': 0.8333101434101585},
  'mlp_relu': {'accuracy': 0.9656841179058513,
   'f1_macro': 0.6715123173153885,
   'roc_auc': 0.8695773464298998},
  'mlp_tanh': {'accuracy': 0.9634843818741751,
   'f1_macro': 0.6638370814646133,
   'roc_auc': 0.8853903483998954}})
```

In [ ]:
```
!brew install --cask mactex
```

==> **Auto-updating Homebrew...**
Adjust how often this is run with HOMEBREW_AUTO_UPDATE_SECS or disable with
HOMEBREW_NO_AUTO_UPDATE. Hide these hints with HOMEBREW_NO_ENV_HINTS (see `m
an brew`).
==> **Downloading https://ghcr.io/v2/homebrew/portable-ruby/portable-ruby/blob
s/sha256:20fa657858e44a4b39171d6e4111f8a9716eb62a78ebbd1491d94f90bb7b830a**
10
0.0%
==> **Pouring portable-ruby-3.4.5.arm64_big_sur.bottle.tar.gz**
 ==> **Homebrew collects anonymous analytics.**
Read the analytics documentation (and how to opt-out) here:
  https://docs.brew.sh/Analytics
No analytics have been recorded yet (nor will be during this `brew` run).

==> **Homebrew is run entirely by unpaid volunteers. Please consider donating:**
  https://github.com/Homebrew/brew#donations

==> **Auto-updated Homebrew!**
Updated 2 taps (homebrew/core and homebrew/cask).
==> **New Formulae**
abpoa: SIMD-based C library for fast partial order alignment using adaptive
band
act_runner: Action runner for Gitea based on Gitea's fork of act
add-determinism: Build postprocessor to reset metadata fields for build repr
oducibility
addlicense: Scan directories recursively to ensure source files have license
headers
addons-linter: Firefox Add-ons linter, written in JavaScript
aiac: Artificial Intelligence Infrastructure-as-Code Generator
aiken: Modern smart contract platform for Cardano
air: Fast and opinionated formatter for R code
alejandra: Command-line tool for formatting Nix Code
anchor: Solana Program Framework
anyzig: Universal zig executable that runs any version of zig
apache-polaris: Interoperable, open source catalog for Apache Iceberg
api-linter: Linter for APIs defined in protocol buffers
apigeecli: Apigee management API command-line interface
arp-scan-rs: ARP scan tool written in Rust for fast local network scans
assimp@5: Portable library for importing many well-known 3D model formats
audiowaveform: Generate waveform data and render waveform images from audio
files
autocycler: Tool for generating consensus long-read assemblies for bacterial
genomes
aws-lc: General-purpose cryptographic library
ayatana-ido: Ayatana Indicator Display Objects
b4: Tool to work with public-inbox and patch archives
backgroundremover: Remove background from images and video using AI
backlog-md: Markdown-native Task Manager & Kanban visualizer for any Git rep
ository
badread: Long read simulator that can imitate many types of read problems
bb-cli: Bitbucket Rest API CLI written in pure PHP
benchi: Benchmarking tool for data pipelines
bento: Fancy stream processing made operationally mundane
bkmr: Unified CLI Tool for Bookmark, Snippet, and Knowledge Management
blueprint-compiler: Markup language and compiler for GTK 4 user interfaces
boa: Embeddable and experimental Javascript engine written in Rust

bom: Utility to generate SPDX-compliant Bill of Materials manifests
bower-mail: Curses terminal client for the Notmuch email system
brename: Cross-platform command-line tool for safe batch renaming via regula
r expressions
breseq: Computational pipeline for finding mutations in short-read DNA reseq
uencing data
brush: Bourne RUsty SHell (command interpreter)
bsc: Bluespec Compiler (BSC)
bstring: Fork of Paul Hsieh's Better String Library
btcli: Bittensor command-line tool
buffrs: Modern protobuf package management
bunster: Compile shell scripts to static binaries
burrow: Kafka Consumer Lag Checking
caesiumclt: Fast and efficient lossy and/or lossless image compression tool
cai: CLI tool for prompting LLMs
camlpdf: OCaml library for reading, writing and modifying PDF files
cargo-careful: Execute Rust code carefully, with extra checking along the wa
y
cargo-clone: Cargo subcommand to fetch the source code of a Rust crate
cargo-component: Create WebAssembly components based on the component model
proposal
cargo-geiger: Detects usage of unsafe Rust in a Rust crate and its dependenc
ies
cargo-shear: Detect and remove unused dependencies from `Cargo.toml` in Rust
projects
ccusage: CLI tool for analyzing Claude Code usage from local JSONL files
changelogen: Generate Beautiful Changelogs using Conventional Commits
chart-releaser: Hosting Helm Charts via GitHub Pages and Releases
chawan: TUI web browser with CSS, inline image and JavaScript support
clang-include-graph: Simple tool for visualizing and analyzing C/C++ project
include graph
claude-squad: Manage multiple AI agents like Claude Code, Aider and Codex in
your terminal
claudekit: Intelligent guardrails and workflow automation for Claude Code
clipper2: Polygon clipping and offsetting library
codex: OpenAI's coding agent that runs in your terminal
cogapp: Small bits of Python computation for static files
concurrentqueue: Fast multi-producer, multi-consumer lock-free concurrent qu
eue for C++11
cookcli: CLI-tool for cooking recipes formated using Cooklang
copyparty: Portable file server
cornelis: Neovim support for Agda
cpdf: PDF Command-line Tools
cram: Functional testing framework for command-line applications
crd2pulumi: Generate typed CustomResources from a Kubernetes CustomResourceD
efinition
credo: Static code analysis tool for the Elixir
dagu: Lightweight and powerful workflow engine
damask-grid: Grid solver of DAMASK - Multi-physics crystal plasticity simula
tion package
darker: Apply Black formatting only in regions changed since last commit
dblab: Database client every command-line junkie deserves
deck: Creates slide deck using Markdown and Google Slides
decker: HyperCard-like multimedia sketchpad
desed: Debugger for Sed
dexidp: OpenID Connect Identity and OAuth 2.0 Provider

diagram: CLI app to convert ASCII arts into hand drawn diagrams
dnglab: Camera RAW to DNG file format converter
docker-debug: Use new container attach on already container go on debug
dockerfmt: Dockerfile format and parser. a modern dockfmt
domain-check: CLI tool for checking domain availability using RDAP and WHOIS protocols
doxx: Terminal document viewer for .docx files
dstp: Run common networking tests against your site
dumbpipe: Unix pipes between devices
dvisvgm: Fast DVI to SVG converter
e2b: CLI to manage E2B sandboxes and templates
eask-cli: CLI for building, running, testing, and managing your Emacs Lisp dependencies
electric: Real-time sync for Postgres
elf2uf2-rs: Convert ELF files to UF2 for USB Flashing Bootloaders
elfio: Header-only C++ library for reading and generating ELF files
emmylua_ls: Lua Language Server
endlessh: SSH tarpit that slowly sends an endless banner
entt: Fast and reliable entity-component system for C++
erlang@27: Programming language for highly scalable real-time systems
erlfmt: Automated code formatter for Erlang
errcheck: Finds silently ignored errors in Go code
execline: Interpreter-less scripting language
execstack: Utility to set/clear/query executable stack bit
faceprints: Detect and label images of faces using local Vision.framework models
fakesteak: ASCII Matrix-like steak demo
fastga: Pairwise whole genome aligner
fastk: K-mer counter for high-fidelity shotgun datasets
fedify: CLI toolchain for Fedify
ferron: Fast, memory-safe web server written in Rust
ffmate: FFmpeg automation layer
ffmpeg@7: Play, record, convert, and stream audio and video
filen-cli: Interface with Filen, an end-to-end encrypted cloud storage service
flexget: Multipurpose automation tool for content
flip-link: Adds zero-cost stack overflow protection to your embedded programs
flye: De novo assembler for single molecule sequencing reads using repeat graphs
forgejo: Self-hosted lightweight software forge
foxglove-cli: Foxglove command-line tool
ftxui: C++ Functional Terminal User Interface
fx-upscale: Metal-powered video upscaling
gbox: Provides environments for AI Agents to operate computer and mobile devices
gcc@14: GNU compiler collection
gcli: Portable Git(hub|lab|tea)/Forgejo/Bugzilla CLI tool
gemini-cli: Interact with Google Gemini AI models from the command-line
gerust: Project generator for Rust backend projects
ggc: Modern Git CLI
ghalint: GitHub Actions linter
girara: GTK+3-based user interface library
gitea-mcp-server: Interactive with Gitea instances with MCP
gitlab-release-cli: Toolset to create, retrieve and update releases on GitLab

gitmux: Git status in tmux status bar
glom: Declarative object transformer and formatter, for conglomerating neste
d data
gnome-papers: Document viewer for PDF and other document formats aimed at th
e GNOME desktop
go-librespot: Spotify client
go-passbolt-cli: CLI for passbolt
go-rice: Easily embed resources like HTML, JS, CSS, images, and templates in
Go
go@1.24: Open source programming language to build simple/reliable/efficient
software
gonzo: Log analysis TUI
goodls: CLI tool to download shared files and folders from Google Drive
goshs: Simple, yet feature-rich web server written in Go
gpgmepp: C++ bindings for gpgme
gpgmepy: Python bindings for gpgme
gphotos-uploader-cli: Command-line tool to mass upload media folders to Goog
le Photos
gradle@8: Open-source build automation tool based on the Groovy and Kotlin D
SL
gravitino: High-performance, geo-distributed, and federated metadata lake
gtrash: Featureful Trash CLI manager: alternative to rm and trash-cli
guichan: Small, efficient C++ GUI library designed for games
hdr10plus_tool: CLI utility to work with HDR10+ in HEVC files
hellwal: Fast, extensible color palette generator
hexd: Colourful, human-friendly hexdump tool
hierarchy-builder: High level commands to declare a hierarchy based on packe
d classes
htmlhint: Static code analysis tool you need for your HTML
hub-tool: Docker Hub experimental CLI tool
hyper-mcp: MCP server that extends its capabilities through WebAssembly plug
ins
ifopt: Light-weight C++ Interface to Nonlinear Programming Solvers
imagineer: Image processing and conversion from the terminal
infat: Tool to set default openers for file formats and url schemes on MacOS
influxdb@2: Time series, events, and metrics database
intermodal: Command-line utility for BitTorrent torrent file creation, verif
ication, etc.
jjui: TUI for interacting with the Jujutsu version control system
jq-lsp: Jq language server
jwt-hack: JSON Web Token Hack Toolkit
kafkactl-aws-plugin: AWS Plugin for kafkactl
kafkactl-azure-plugin: Azure Plugin for kafkactl
kargo: Multi-Stage GitOps Continuous Promotion
kbt: Keyboard tester in terminal
kingfisher: MongoDB's blazingly fast secret scanning and validation tool
kissat: Bare metal SAT solver
kokkos: C++ Performance Portability Ecosystem for parallel execution and abs
traction
kraken2: Taxonomic sequence classification system
ktop: Top-like tool for your Kubernetes clusters
kubectl-ai: AI powered Kubernetes Assistant
kubernetes-cli@1.32: Kubernetes command-line interface
kubernetes-mcp-server: MCP server for Kubernetes
lakekeeper: Apache Iceberg REST Catalog
lazycontainer: Terminal UI for Apple Containers

ldcli: CLI for managing LaunchDarkly feature flags
libayatana-appindicator: Ayatana Application Indicators Shared Library
libayatana-indicator: Ayatana Indicators Shared Library
libbsc: High performance block-sorting data compression library
libcaption: Free open-source CEA608 / CEA708 closed-caption encoder/decoder
libdatrie: Double-Array Trie Library
libdbusmenu: GLib and Gtk Implementation of the DBusMenu protocol
libjodycode: Shared code used by several utilities written by Jody Bruchon
libngtcp2: IETF QUIC protocol implementation
libpq@16: Postgres C API library
libudfread: Universal Disk Format reader
lima-additional-guestagents: Additional guest agents for Lima
limine: Modern, advanced, portable, multiprotocol bootloader and boot manage
r
lld@20: LLVM Project Linker
llvm@20: Next-gen compiler infrastructure
lnk: Git-native dotfiles management that doesn't suck
lolcrab: Make your console colorful, with OpenSimplex noise
lsr: Ls but with io_uring
lstr: Fast, minimalist directory tree viewer
lunarml: Standard ML compiler that produces Lua/JavaScript
lunasvg: SVG rendering and manipulation library in C++
lutgen: Blazingly fast interpolated LUT generator and applicator for color p
alettes
lzsa: Lossless packer that is optimized for fast decompression on 8-bit micr
os
mac-cleanup-py: Python cleanup script for macOS
manifold: Geometry library for topological robustness
mcp-get: CLI for discovering, installing, and managing MCP servers
mcp-inspector: Visual testing tool for MCP servers
mcp-proxy: Bridge between Streamable HTTP and stdio MCP transports
mcphost: CLI host for LLMs to interact with tools via MCP
mcptools: CLI for interacting with MCP servers using both stdio and HTTP tra
nsport
media-control: Control and observe media playback from the command-line
melt: Backup and restore Ed25519 SSH keys with seed words
memtier_benchmark: Redis and Memcache traffic generation and benchmarking to
ol
mender-artifact: CLI tool for managing Mender artifact files
mender-cli: General-purpose CLI tool for the Mender backend
mermaid-cli: CLI for Mermaid library
min-lang: Small but practical concatenative programming language and shell
miniflux: Minimalist and opinionated feed reader
minify: Minifier for HTML, CSS, JS, JSON, SVG, and XML
miniprot: Align proteins to genomes with splicing and frameshift
mk: Wrapper for auto-detecting build and test commands in a repository
mklittlefs: Creates LittleFS images for ESP8266, ESP32, Pico RP2040, and RP2
350
mlc: Check for broken links in markup files
mongo-c-driver@1: C driver for MongoDB
moodle-dl: Downloads course content fast from Moodle (e.g., lecture PDFs)
moribito: TUI for LDAP Viewing/Queries
mpremote: Tool for interacting remotely with MicroPython devices
msolve: Library for Polynomial System Solving through Algebraic Methods
multi-gitter: Update multiple repositories in with one command
nanoarrow: Helpers for Arrow C Data & Arrow C Stream interfaces

nelm: Kubernetes deployment tool that manages and deploys Helm Charts
nerdlog: TUI log viewer with timeline histogram and no central server
nessie: Transactional Catalog for Data Lakes with Git-like semantics
netscanner: Network scanner with features like WiFi scanning, packetdump and more
newsraft: Terminal feed reader
nextflow: Reproducible scientific workflows
nifi-toolkit: Command-line utilities to setup and support NiFi
nip4: Image processing spreadsheet
nixfmt: Command-line tool to format Nix language code
notion-mcp-server: MCP Server for Notion
npq: Audit npm packages before you install them
nuxi: Nuxt CLI (nuxi) for creating and managing Nuxt projects
nx: Smart, Fast and Extensible Build System
nyan: Colorizing `cat` command with syntax highlighting
oasis: CLI for interacting with the Oasis Protocol network
omekasy: Converts alphanumeric input to various Unicode styles
omnara: Talk to Your AI Agents from Anywhere
onigmo: Regular expressions library forked from Oniguruma
openapi: CLI tools for working with OpenAPI, Arazzo and Overlay specificatio ns
opkssh: Enables SSH to be used with OpenID Connect
osx-trash: Allows trashing of files instead of tempting fate with rm
oterm: Terminal client for Ollama
otterdog: Manage GitHub organizations at scale using an infrastructure as co de approach
ovsx: Command-line interface for Eclipse Open VSX
oxen: Data VCS for structured and unstructured machine learning datasets
pangene: Construct pangenome gene graphs
pdtm: ProjectDiscovery's Open Source Tool Manager
perbase: Fast and correct perbase BAM/CRAM analysis
pg-schema-diff: Diff Postgres schemas and generating SQL migrations
pgslice: Postgres partitioning as easy as pie
pieces-cli: Command-line tool for Pieces.app
pixd: Visual binary data using a colour palette
plakar: Create backups with compression, encryption and deduplication
plutobook: Paged HTML Rendering Library
plutoprint: Generate PDFs and Images from HTML
plutovg: Tiny 2D vector graphics library in C
podcast-archiver: Archive all episodes from your favorite podcasts
polaris: Validation of best practices in your Kubernetes clusters
policy-engine: Unified Policy Engine
polypolish: Short-read polishing tool for long-read assemblies
preevy: Quickly deploy preview environments to the cloud
prog8: Compiled programming language targeting the 8-bit 6502 CPU family
protoc-gen-doc: Documentation generator plugin for Google Protocol Buffers
protozero: Minimalist protocol buffer decoder and encoder in C++
pulp-cli: Command-line interface for Pulp 3
pulumictl: Swiss army knife for Pulumi development
pyp: Easily run Python at the shell! Magical, but never mysterious
pyrefly: Fast type checker and IDE for Python
pytr: Use TradeRepublic in terminal and mass download all documents
qman: Modern man page viewer
qnm: CLI for querying the node_modules directory
qrkey: Generate and recover QR codes from files for offline private key back up

quadcastrgb: Set RGB lights on HyperX QuadCast S and Duocast microphones
qwen-code: AI-powered command-line workflow tool for developers
rasusa: Randomly subsample sequencing reads or alignments
readerwriterqueue: Fast single-producer, single-consumer lock-free queue for C++
readsb: ADS-B decoder swiss knife
reckoner: Declaratively install and manage multiple Helm chart releases
rggen: Code generation tool for control and status registers
rmpc: Terminal based Media Player Client with album art support
rna-star: RNA-seq aligner
rnp: High performance C++ OpenPGP library used by Mozilla Thunderbird
rocq-elpi: Elpi extension language for Rocq
rofi: Window switcher, application launcher and dmenu replacement
ropebwt3: BWT construction and search
rqbit: Fast command-line bittorrent client and server
rsql: CLI for relational databases and common data file formats
rulesync: Unified AI rules management CLI tool
rv: Ruby version manager
s6-rc: Process supervision suite
sacad: Automatic cover art downloader
samply: CLI sampling profiler
scdl: Command-line tool to download music from SoundCloud
secretspec: Declarative secrets management tool
seqan3: Modern C++ library for sequence analysis
sequoia-chameleon-gnupg: Reimplementatilon of gpg and gpgv using Sequoia
sexpect: Expect for shells
shamrock: Astrophysical hydrodynamics using SYCL
sherif: Opinionated, zero-config linter for JavaScript monorepos
skalibs: Skarnet's library collection
skani: Fast, robust ANI and aligned fraction for (metagenomic) genomes and contigs
smenu: Powerful and versatile CLI selection tool for interactive or scripting use
somo: Human-friendly alternative to netstat for socket and port monitoring
sox_ng: Sound eXchange NG
specify: Toolkit to help you get started with Spec-Driven Development
spice-server: Implements the server side of the SPICE protocol
sprocket: Bioinformatics workflow engine built on the Workflow Description Language (WDL)
sqlite-rsync: SQLite remote copy tool
sqruff: Fast SQL formatter/linter
standardebooks: Tools for producing ebook files
stormy: Minimal, customizable and neofetch-like weather CLI based on rainy
stringtie: Transcript assembly and quantification for RNA-Seq
stu: TUI explorer application for Amazon S3 (AWS S3)
style-dictionary: Build system for creating cross-platform styles
supabase-mcp-server: MCP Server for Supabase
swift-section: CLI tool for parsing mach-o files to obtain Swift information
swiftly: Swift toolchain installer and manager
sylph: Ultrafast taxonomic profiling and genome querying for metagenomic samples
tabixpp: C++ wrapper to tabix indexer
tdd-guard: Automated TDD enforcement for Claude Code
tdom: XML/DOM/XPath/XSLT/HTML/JSON implementation for Tcl
technitium-dns: Self host a DNS server for privacy & security
technitium-library: Library for technitium .net based applications

terraform-mcp-server: MCP server for Terraform
terraform-module-versions: CLI that checks Terraform code for module updates
teslamate: Self-hosted data logger for your Tesla
tfcmt: Notify the execution result of terraform command
tfmcp: Terraform Model Context Protocol (MCP) Tool
tfmv: CLI to rename Terraform resources and generate moved blocks
tfsort: CLI to sort Terraform variables and outputs
tiledb: Universal storage engine
timoni: Package manager for Kubernetes, powered by CUE and inspired by Helm
tiny-remapper: Tiny, efficient tool for remapping JAR files using "Tiny"-for
mat mappings
tldx: Domain Availability Research Tool
tmex: Minimalist tmux layout manager
tmuxai: AI-powered, non-intrusive terminal assistant
tofu-ls: OpenTofu Language Server
toml-bombadil: Dotfile manager with templating
tree-sitter-cli: Parser generator tool
trimal: Automated alignment trimming in large-scale phylogenetic analyses
tsnet-serve: Expose HTTP applications to a Tailscale Tailnet network
tsui: TUI for configuring and monitoring Tailscale
tun2proxy: Tunnel (TUN) interface for SOCKS and HTTP proxies
twitch-cli: CLI to make developing on Twitch easier
two-ms: Detect secrets in files and communication platforms
typtea: Minimal terminal-based typing speed tester
uhubctl: USB hub per-port power control
undercutf1: F1 Live Timing TUI for all F1 sessions with variable delay to sy
nc to your TV
unitycatalog: Open, Multi-modal Catalog for Data & AI
unoserver: Server for file conversions with Libre Office
urx: Extracts URLs from OSINT Archives for Security Insights
uvwasi: WASI syscall API built atop libuv
varlock: Add declarative schema to .env files using @env-spec decorator comm
ents
videoalchemy: Toolkit expanding video processing capabilities
vineflower: Java decompiler
vsd: Download video streams over HTTP, DASH (.mpd), and HLS (.m3u8)
wait4x: Wait for a port or a service to enter the requested state
wayback: Archiving tool integrated with various archival services
webdav: Simple and standalone WebDAV server
wgpu-native: Native WebGPU implementation based on wgpu-core
wishlist: Single entrypoint for multiple SSH endpoints
wrkflw: Validate and execute GitHub Actions workflows locally
wxwidgets@3.2: Cross-platform C++ GUI toolkit
xml2rfc: Tool to convert XML RFC7749 to the original ASCII or the new HTML l
ook-and-feel
xtl: X template library
yaml2json: Command-line tool convert from YAML to JSON
yeet: Packaging tool that lets you declare build instructions in JavaScript
yek: Fast Rust based tool to serialize text-based files for LLM consumption
zig@0.14: Programming language designed for robustness, optimality, and clar
ity
zsh-history-enquirer: Zsh plugin that enhances history search interaction

You have **7** outdated formulae installed.

==> **Caveats**

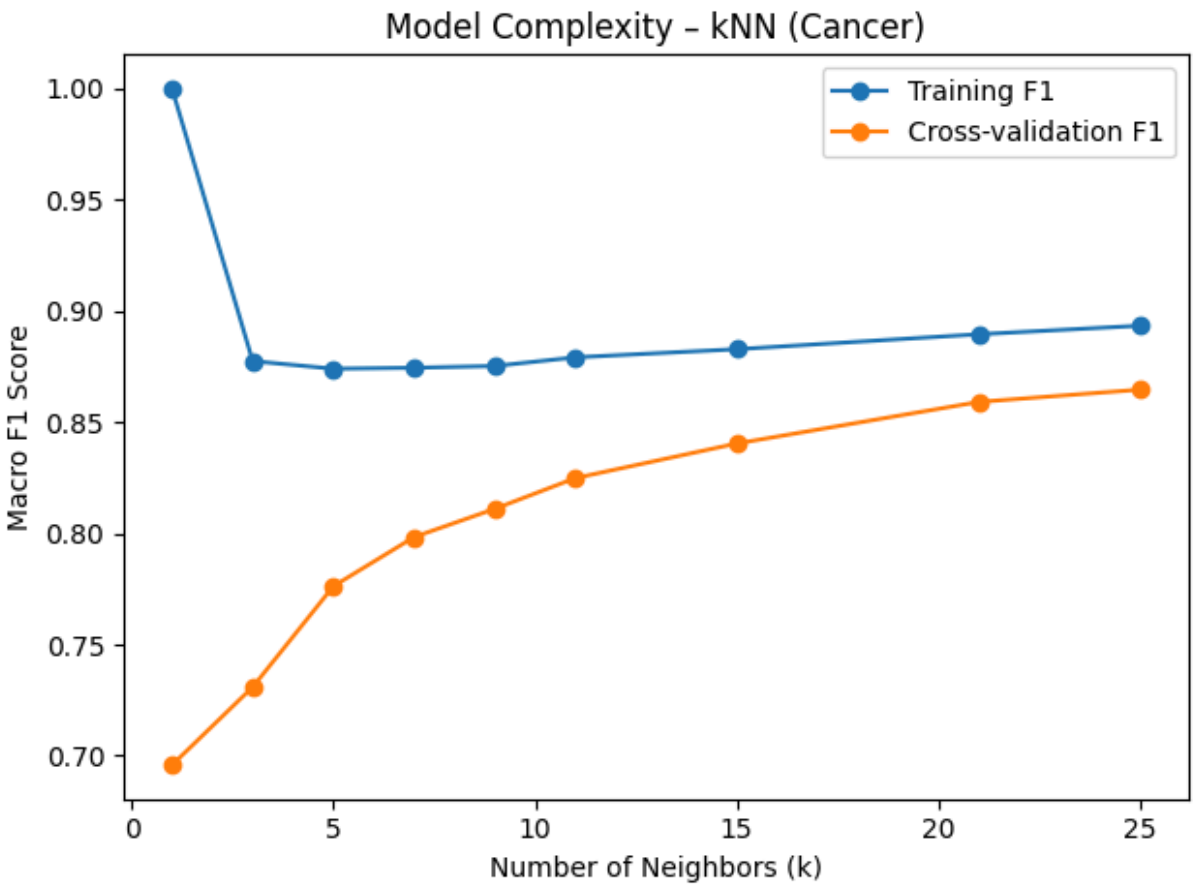
```
You must restart your terminal window for the installation of MacTeX CLI
tools to take effect.

Alternatively, Bash and Zsh users can run the command:

  eval "$(/usr/libexec/path_helper)"

==> Downloading https://mirror.ctan.org/systems/mac/mactex/mactex-20250308.p
kg
==> Downloading from https://ctan.math.washington.edu/tex-archive/systems/ma
c/ma
##############                                                        1
9.2%
```

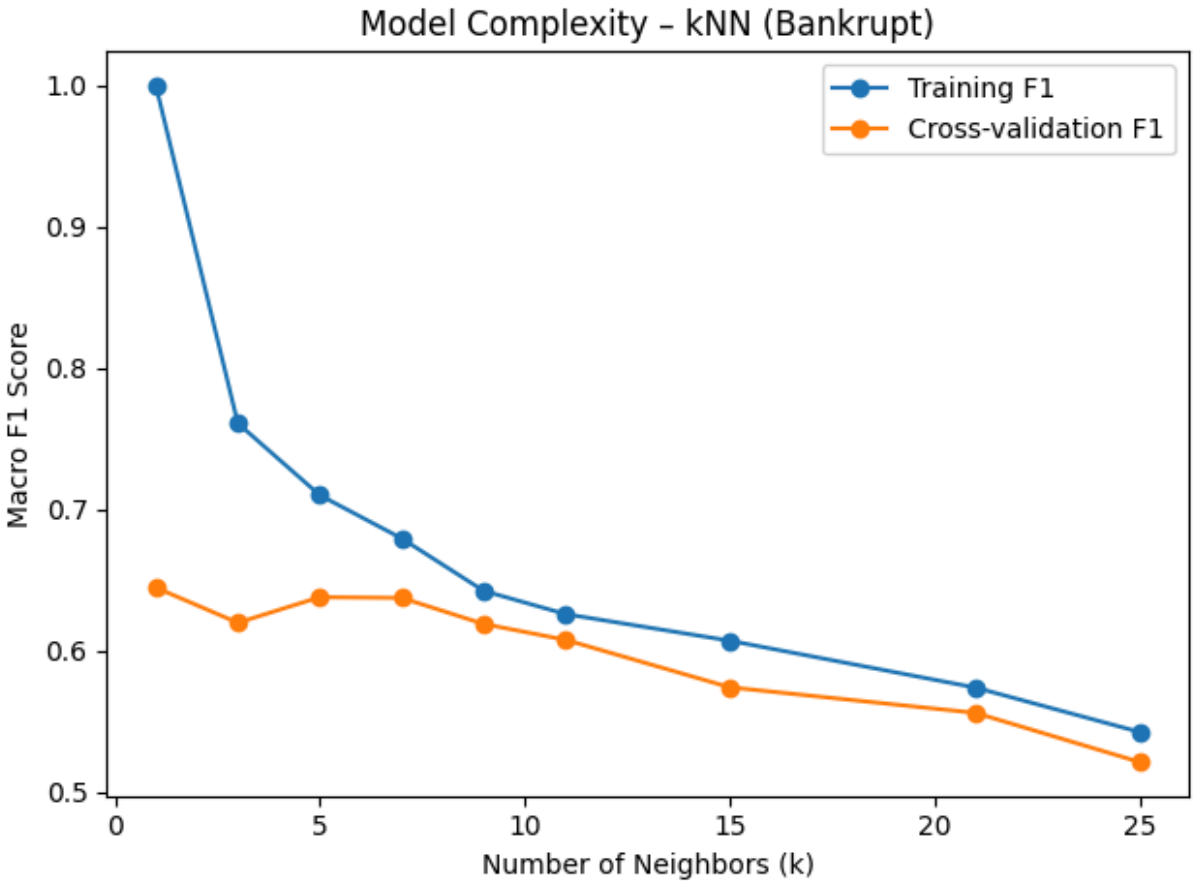
In [ ]:
```
!xelatex --version
```

In [12]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.model_selection import StratifiedKFold, validation_curve
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import f1_score, make_scorer

def plot_knn_complexity_curve(df, dataset="cancer", target_col=None, save_pa
    """
    Plots kNN model complexity curve (neighbors vs. F1 score) for cancer_df
    """

    # Step 1. Target + Features
    if dataset == "cancer":
        if "Patient_ID" in df.columns:
            df = df.drop(columns=["Patient_ID"])
        if target_col is None:
            target_col = "Target_Severity_Score"
        df["SeverityLevel"] = pd.qcut(df[target_col], q=3, labels=[0,1,2])
        y = df["SeverityLevel"]
        X = df.drop(columns=[target_col, "SeverityLevel"])
    elif dataset == "bankrupt":
        if target_col is None:
            target_col = "Bankrupt?"
        y = df[target_col]
        X = df.drop(columns=[target_col])
    else:
        raise ValueError("dataset must be 'cancer' or 'bankrupt'")

    # Step 2. Preprocessing
    if dataset == "cancer":
        categorical_cols = X.select_dtypes(include=["object", "category"]).c
        numeric_cols = X.select_dtypes(include=[np.number]).columns.tolist()
        numeric_tf = Pipeline(steps=[
            ("imputer", SimpleImputer(strategy="median")),
            ("scaler", StandardScaler())
```

```
        ])
        categorical_tf = Pipeline(steps=[
            ("imputer", SimpleImputer(strategy="most_frequent")),
            ("ohe", OneHotEncoder(handle_unknown="ignore"))
        ])
        preprocess = ColumnTransformer(
            transformers=[
                ("num", numeric_tf, numeric_cols),
                ("cat", categorical_tf, categorical_cols),
            ]
        )
    else:  # bankrupt = all numeric
        preprocess = Pipeline(steps=[
            ("imputer", SimpleImputer(strategy="median")),
            ("scaler", StandardScaler())
        ])

    # Step 3. kNN pipeline
    pipe = Pipeline(steps=[
        ("prep", preprocess),
        ("knn", KNeighborsClassifier())
    ])

    # Step 4. Validation curve
    param_range = [1, 3, 5, 7, 9, 11, 15, 21, 25]
    cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
    scorer = make_scorer(f1_score, average="macro")

    train_scores, test_scores = validation_curve(
        pipe, X, y,
        param_name="knn__n_neighbors",
        param_range=param_range,
        cv=cv, scoring=scorer, n_jobs=-1
    )

    train_mean = train_scores.mean(axis=1)
    test_mean = test_scores.mean(axis=1)

    # Step 5. Plot
    plt.figure()
    plt.plot(param_range, train_mean, marker="o", label="Training F1")
    plt.plot(param_range, test_mean, marker="o", label="Cross-validation F1"
    plt.xlabel("Number of Neighbors (k)")
    plt.ylabel("Macro F1 Score")
    plt.title(f"Model Complexity — kNN ({dataset.capitalize()})")
    plt.legend()
    plt.tight_layout()

    if save_path:
        plt.savefig(save_path, dpi=200)
    plt.show()


# --- Run for Cancer dataset ---
plot_knn_complexity_curve(cancer_df, dataset="cancer", save_path="download-k
```

```
# --- Run for Bankruptcy dataset ---
plot_knn_complexity_curve(bankrupt_df, dataset="bankrupt", save_path="downlo
```




Model Complexity – kNN (Bankrupt)
Training F1
Cross-validation F1
Macro F1 Score
1.0
0.9
0.8
0.7
0.6
0.5
0
5
10
15
20
25
Number of Neighbors (k)

In [ ]: